



❖ 图 7.4 文字双层阴影



❖ 图 7.5 文字层叠效果



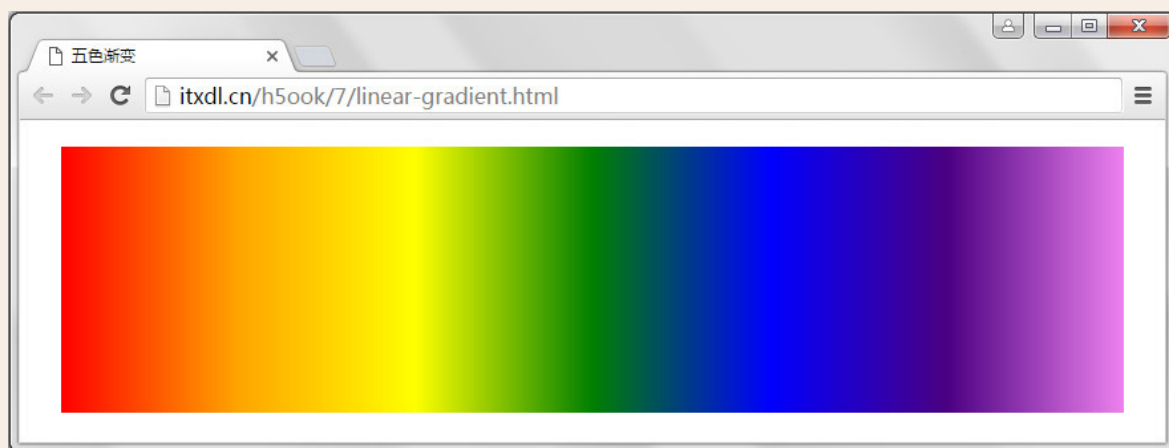
❖ 图 7.6 文字光晕效果



❖ 图 7.7 火焰文字效果

VISIT...

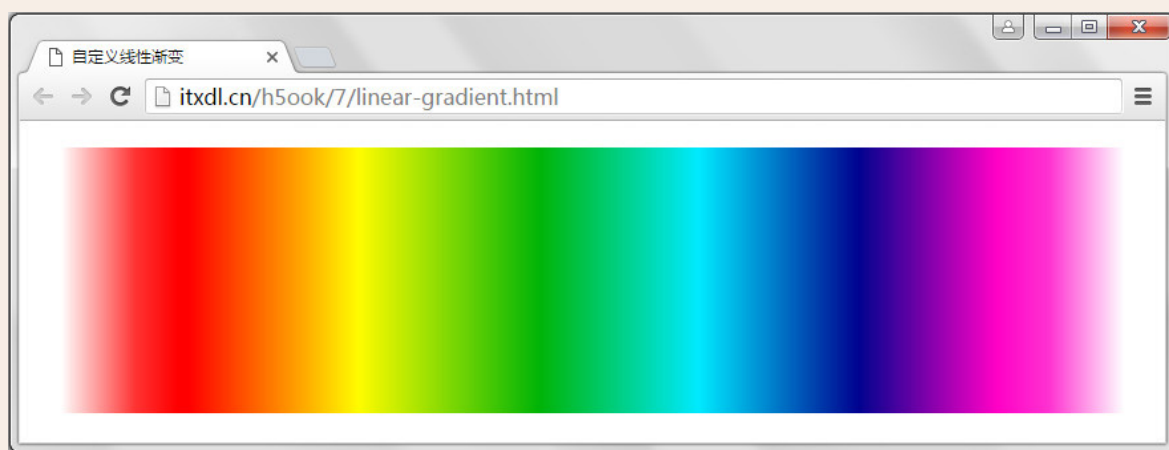
LANZAROTE
Caliente.COM



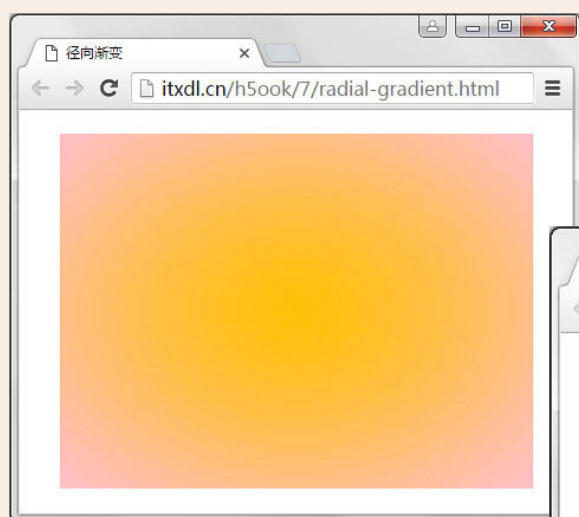
❖ 图 7.59 多色线性渐变



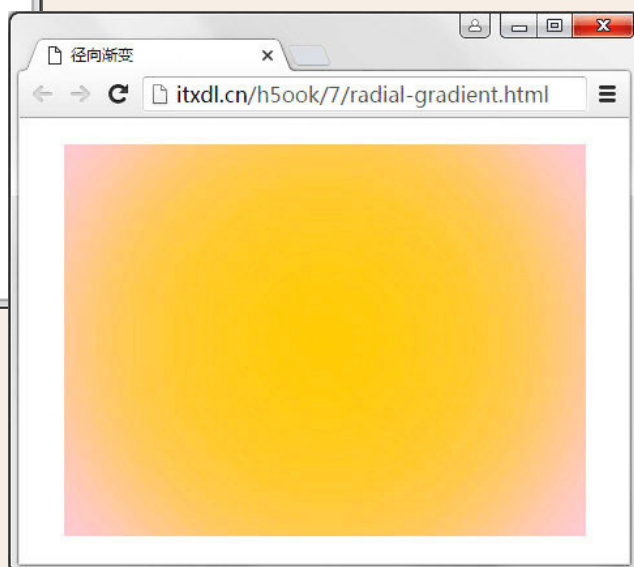
❖ 图 7.60 自定义线性渐变(1)



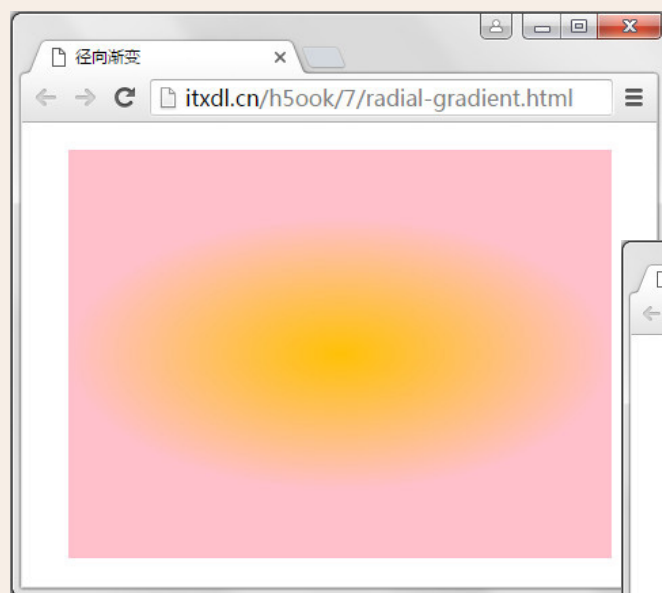
❖ 图 7.61 自定义线性渐变(2)



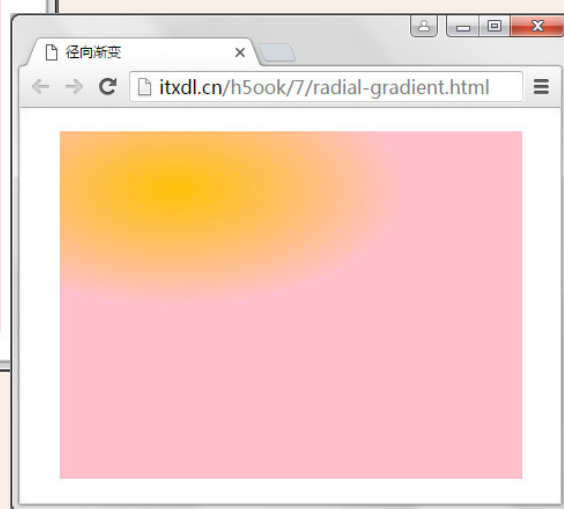
❖ 图 7.63 简单径向渐变



❖ 图 7.64 圆形渐变



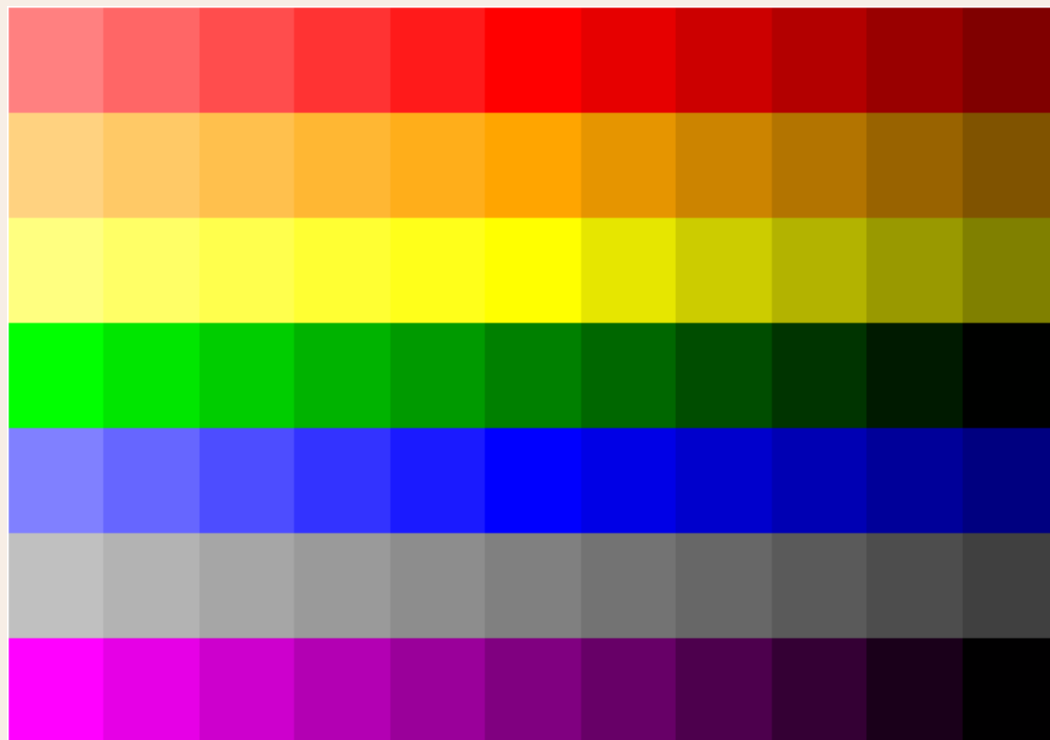
❖ 图 7.65 自定义半径的径向渐变



❖ 图 7.66 自定义半径及圆心位置的径向渐变



❖ 图 7.67 三色重复的径向渐变



❖ 图 10.10 七色板示意图

细说网页制作

兄弟连教育◎组编

高洛峰 李明霞 王宝龙◎编著

電子工業出版社·

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

通过本书的学习，能快速上手网站前端开发。为了使 HTML5 语言能被读者更好地掌握和应用，同时作为“跟兄弟连学 HTML5 系列教程”的第一本入门级图书，本书对 HTML 语言的知识点进行了详细的阐述和分析，包括 HTML5 和 CSS3 的语法、各种页面布局方法、流行的前端框架 Bootstrap 等内容。书中不仅有通俗易懂的语法讲解，也用贴切的小案例实验，使读者能轻松掌握新知识，且可以快速上手前端技术操作。本书适合对前端技术开发感兴趣的初学者阅读，也可以作为从事前端技术工作的开发人员的参考书，或作为大学生学习 HTML5 的教材。同时，本书也为读者之后学习同系列其他图书奠定了基础。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

细说网页制作 / 兄弟连教育组编；高洛峰，李明霞，王宝龙编著. —北京：电子工业出版社，2017.11

ISBN 978-7-121-32875-6

I. ①细… II. ①兄… ②高… ③李… ④王… III. ①网页制作工具—高等学校—教材 IV. ①TP393.092.2

中国版本图书馆 CIP 数据核字（2017）第 247282 号

策划编辑：李 冰

责任编辑：李 冰

特约编辑：赵海红 赵海军等

印 刷：三河市鑫金马印装有限公司

装 订：三河市鑫金马印装有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×1092 1/16 印张：25.5 字数：659.2 千字 彩插：2

版 次：2017 年 11 月第 1 版

印 次：2017 年 11 月第 1 次印刷

定 价：59.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：libing@phei.com.cn。

前言

PREFACE

随着 HTML5 标准化逐渐成熟，以及互联网的飞速发展和移动端的应用不断创新，再加上微信公众号、小程序的应用飙升，原生 APP 向 Web APP 和混合 APP 的转变，用户对视觉效果和操作体验的要求越来越高，HTML5 成为移动互联网的主要技术，也是目前的主流技术之一。HTML5 是超文本标记语言（HTML）的第 5 次修订，是近年来 Web 标准的巨大飞跃。Web 是一个内涵极为丰富的平台，和以前版本不同的是，HTML5 并非仅仅用来表示 Web 内容，在这个平台上还能非常方便地加入视频、音频、图像、动画，以及与计算机的交互。HTML5 的意义在于它带来了一个无缝的网络，无论是 PC、平板电脑，还是智能手机，都能非常方便地浏览基于 HTML5 的各类网站。对用户来说，手机上的 APP 会越来越少，用 HTML5 实现的一些应用不需要下载安装，就能立即在手机界面中生成一个 APP 图标，使用手机中的浏览器来运行，新增的导航标签也能更好地帮助小屏幕设备和有视力障碍人士使用。HTML5 拥有服务器推送技术，给用户带来了更便捷的实时聊天功能和更快速的网游体验。

HTML5 对于开发者来说更是福音。HTML5 本身是由 W3C 推荐的，也就意味着每一个浏览器或每一个平台都可以实现，这样可以节省开发者花在浏览器页面展现兼容性上的时间。另外，HTML5 是 Web 前端技术的一个代名词，其核心技术点还是 JavaScript。如 HTML5 的服务器推送技术再结合 JavaScript 编程，能够帮助我们实现服务器将数据“推送”到客户端的功能，客户端与服务器之间的数据传输将更加高效。基于 SVG、Canvas、WebGL 及 CSS3 的 3D 功能，会让用户惊叹在浏览器中所呈现的各种炫酷的视觉效果。以往在 iPhone iPad 上不支持的 Flash 将来都有可能通过 HTML5 华丽地呈现在用户的 iOS 设备上。

本套图书介绍

为了让前端技术初学者少走弯路，快速而轻松地学习 HTML5 和 JavaScript 编程，我们结合新技术和兄弟连多年的教学经验积累，再通过对企业实际应用的调研，编写了一整套 HTML5 系列图书，共 5 本，包括《细说网页制作》、《细说 JavaScript 语言》、《细说 DOM 编程》、《细说 AJAX 与 jQuery》和《细说 HTML5 高级 API》。每一本书都是不同层次的完整



内容，不仅给初学者安排了循序渐进的学习过程，也便于不同层次的读者选择；既适合没有编程基础的前端技术初学者作为入门教程，也适合正在从事前端开发的人员作为技术提升参考资料。本套图书编写的初衷是为了紧跟新技术和兄弟连 IT 教育 HTML5 学科的教学发展，作为本校培训教程使用，也可作为大、中专院校和其他培训学校的教材。同时，对于前端开发爱好者，本书也有较高的参考价值。

《细说网页制作》

作为“跟兄弟连学 HTML5 系列教程”的第一本书，主要带领 HTML5 初学者一步步完成精美的页面制作。本书内容包括 HTML 应用、CSS 应用、HTML5 的新技术、各种主流的页面布局方法和一整套页面开发实战技能，让读者可以使用多种方法完成 PC 端的页面制作、移动端的页面制作，以及响应式布局页面的制作，不仅能做出页面，还能掌握如何做好页面。

《细说JavaScript语言》

这是“跟兄弟连学 HTML5 系列教程”的第二本书，在学习本书之前需要简单了解一下第一本书中的 HTML 和 CSS 内容。本书内容是纯 JavaScript 语言部分，和浏览器无关，包括 JavaScript 基本语法、数据类型、流程控制、函数、对象、数组和内置对象，所有知识点都是为了学习 DOM 编程、Node.js、JS 框架等 JavaScript 高级部分做准备。本书虽然是 JavaScript 的基础部分，但全书内容都需要牢牢掌握，才能更好地晋级学习。

《细说DOM编程》

这是“跟兄弟连学 HTML5 系列教程”的第三本书，全书内容都和浏览器相关，在学习本书之前需要掌握前两本书的技术。本书内容包括 BOM 和 DOM 两个关键技术点，并且全部以 PC 端和移动端的 Web 特效为主线，以实例贯穿全部知识点进行讲解。学完本书的内容，不仅可以用 JavaScript 原生的语法完成页面的特效编写，也为学习后面的 JavaScript 框架课程做好了准备。本书内容是 Web 前端课程的核心，需要读者按书中的实例多加练习，能熟练地进行浏览器中各种特效程序的开发。

《细说AJAX与jQuery》

这是“跟兄弟连学 HTML5 系列教程”的第四本书，其内容是建立在第三本书之上的，包括服务器端开发语言 Node.js、异步传输 AJAX 和 jQuery 框架三部分。其中，Node.js 部分是为了配合 AJAX 完成客户端向服务器端的异步请求；jQuery 是目前主流的前端开发框架，

其目的是让开发者用尽量少的代码完成尽可能多的功能。AJAX 和 jQuery 是目前前端开发的必备技术，本书从基本应用开始学起，用实例分解方式讲解技术点，让读者完全掌握这些必备的技能。

《细说HTML5 高级API》

这是“跟兄弟连学 HTML5 系列教程”的第五本书，是前端开发的应用部分，主要讲解 HTML5 高级 API 的相关内容，包括画布、Web 存储、应用缓存、服务器发送事件等，可以用来开发移动端的 Web APP 项目。本书重点讲解了 Cordova 技术，它提供了一组与设备相关的 API，通过这组 API，移动应用就能够通过 JavaScript 访问原生的设备功能，如摄像头、麦克风等。Cordova 还提供了一组统一的 JavaScript 类库，以及与这些类库所用的设备相关的原生后台代码。通过编写 HTML5 程序，再用 Cordova 打包出混合 APP 的项目，可以安装在 Android 和 iOS 等设备上。

本套图书的特点

1. 内容丰富，由浅入深

本套图书在内容组织上本着“起点低，重点高”的原则，内容几乎涵盖前端开发的所有核心技能，对于某一方面的介绍再从多角度进行延伸。为了让读者更加方便地学习本套图书的内容，在每本书的每个章节中都提供了一些实际的项目案例，便于读者在实践中学习。

2. 结构清晰，讲解到位

每个章节都环环相扣，为了让初学者更快地上手，本套图书精心设计了学习方式。对于概念的讲解，都是先用准确的语言总结概括，再用直观的图示演示过程，接着以详细的注释解释代码，最后用形象的比喻帮助记忆。对于框架部分，先提取核心功能快速掌握框架的应用，再用多个对应的实例分别讲解每个模块，最后逐一讲解框架的每个功能。对于代码部分，先演示程序效果，再根据需求总结涉及的知识点逐一讲解，然后组合成实例，最后总结分析重点功能的逻辑实现。

3. 完整案例，代码实用

为了便于读者学习，本套图书的全部案例都可以在商业项目中直接运用，丰富的案例几乎涵盖前端应用的各个方面。所有的案例都可以通过对应的二维码扫描，直接在手机上查看运行结果，读者可以通过仔细研究其效果，最大限度地掌握开发技术。另外，扫描每个章节中的资源下载二维码，可以获得下载链接，点击链接即可获取所有案例的完整源代码。

4. 视频精致，立体学习

字不如表，表不如图，图不如视频，每本书都配有详细讲解的教学视频，由兄弟连名师



精心录制，不仅能覆盖书中的全部知识点，而且远远超出书中的内容。通过参考本套图书，再结合教学视频学习，可以加快对知识点的掌握，加快学习进度。读者可以扫描每个章节中提供的教学视频二维码，获取视频列表直接在手机上观看，也可以直接登录“猿代码（www.ydma.cn）”平台在 PC 端观看，逐步掌握每个技术点。

5. 电子教案，学教通用

每本书都提供了和章节配套的电子教案（PPT）。对于学生来说，电子教案可以作为学习笔记使用，是知识点的浓缩和重点内容的记录。由于本套图书可以作为高校相关课程的教材或课外辅导书，所以可以方便教师教学使用。读者可以通过扫描对应章节的二维码，下载或在线观看电子教案。本书为部分章节提供了一些扩展文章，也可以通过扫描二维码的方式下载或在线观看。

6. 实时测试，寓学于练

每章最后都提供了专门的测试习题，供读者检验所学知识是否牢固掌握。通过扫描测试习题对应的二维码，可以查看答案和详细的讲解。

7. 技术支持，服务到位

为了帮助读者学到更多的 HTML5 技术，在兄弟连论坛（bbs.itxdl.cn）中还可以下载常用的技术手册和所需的软件。笔者及兄弟连 IT 教育（新三板上市公司，股票代码：839467）的全体讲师和技术人员也会及时回答读者的提问，与读者进行在线技术交流，并为读者提供各类技术文章，帮助读者提高开发水平，解决读者在开发中遇到的疑难问题。

本套图书的读者群

- 有审美，喜欢编程，并且怀揣梦想的有志青年。
- 打算进入前端编程大门的新手，阶梯递进，由浅入深。
- 专业培训机构前端课程授课教材，有体系地掌握全部前端技能。
- 各大院校的在校学生和相关的授课老师，课件、试题、代码丰富实用。
- 前端页面、Web APP、网页游戏、微信公众号等开发的前沿程序员，是专业人员的开发工具。
- 其他方向的编程爱好者，需要前端技术配合，或转向前端开发的程序员。

参与本书编写的人员还有李明霞、王宝龙和李明，在此一并表示感谢！

2017 年 8 月

目录

CONTENTS

第 1 章	介绍 HTML5	1
1.1	了解 HTML5 的主流应用	1
1.1.1	表单的强大	2
1.1.2	响应式页面布局	2
1.1.3	与用户交互的特效	4
1.1.4	微网站的制作	5
1.1.5	基于 HTML5 的移动 APP 开发	6
1.1.6	HTML5 游戏	7
1.1.7	多媒体的应用	7
1.2	什么是 HTML5	8
1.2.1	HTML5 和 HTML 的关系	8
1.2.2	HTML 和 CSS 的关系	10
1.2.3	HTML5 和 CSS3 的关系	11
1.2.4	HTML5 和 JavaScript 的关系	11
1.3	HTML5 的靠山	12
1.3.1	W3C 是什么	12
1.3.2	IETF 是什么	13
1.3.3	RFC 是什么	14
1.3.4	WHATWG 是什么	14
1.3.5	Web 的新标准	14
1.4	HTML5 的曲折发展过程	15
1.4.1	HTML5 的诞生	15
1.4.2	浏览器之间的大战	16
1.4.3	HTML5 技术的应用现状	18



1.4.4	HTML5 平台的兴起	18
1.4.5	HTML5 行业的发展预测	20
1.5	HTML5 的学习线路图	21
1.5.1	第一阶段——学习网页制作	23
1.5.2	第二阶段——编写用户交互功能	24
1.5.3	第三阶段——成为前端工程师	25
	本章小结	26
	本章习题	26
第 2 章	HTML5 的基本语法	28
2.1	课前准备	28
2.1.1	了解 Web	29
2.1.2	了解 HTML	30
2.1.3	了解 HTML 运行原理	30
2.1.4	如何选择开发工具	31
2.1.5	认识浏览器中的开发者工具	32
2.1.6	现在学习 HTML5 的方式	34
2.1.7	简单 HTML 实例制作	35
2.2	HTML 语言的语法	36
2.2.1	HTML 标签和元素	36
2.2.2	HTML 语法不区分字母大小写	37
2.2.3	HTML 标签属性	37
2.2.4	HTML 颜色值的设置	37
2.2.5	HTML 文档注释	38
2.2.6	HTML 代码格式	38
2.2.7	HTML 字符实体	38
2.3	HTML 文档的主体结构	39
2.3.1	HTML 文档类型的新定义方式	40
2.3.2	HTML 文档的主体标签	41
2.4	HTML 文档头部标签<head>	41
2.4.1	<title>标签	42
2.4.2	<base>标签	42
2.4.3	<link>标签	43
2.4.4	<meta>标签	43
2.5	HTML 文档主体标签<body>	44
2.6	HTML5 做到了与之前版本的兼容	45
2.6.1	可以省略标记的元素	45

2.6.2 具有 boolean 值的属性	46
2.6.3 引号的使用	46
2.7 设置 IE 9 以下版本浏览器支持 HTML5	46
本章小结	47
本章习题	47
第 3 章 HTML5 文字版面和编辑标签	49
3.1 HTML 基础标签	49
3.1.1 标题标签<h1>~<h6>	50
3.1.2 换行标签 和段落标签<p>	51
3.1.3 水平分隔线标签<hr>	51
3.1.4 格式标签	51
3.1.5 文本标签	53
3.2 使用 HTML 表格	55
3.3 HTML 框架结构	57
本章小结	60
本章习题	61
第 4 章 多媒体应用	62
4.1 创建图像和链接	62
4.1.1 URL 概述	63
4.1.2 插入图片	63
4.1.3 建立锚点和超链接	64
4.2 HTML 图像地图	66
4.2.1 什么是图像地图	66
4.2.2 图像地图如何制作	66
4.2.3 实现图像地图	67
4.3 新增多媒体播放元素	68
本章小结	71
本章习题	71
第 5 章 HTML5 表单	74
5.1 HTML 表单中的 get 和 post 方法	74
5.1.1 get 方法	75
5.1.2 post 方法	76
5.1.3 HTML 表单中 get 和 post 的区别	76



5.2	HTML 表单设计	77
5.2.1	表单标签<form>	77
5.2.2	文本域和密码域	78
5.2.3	提交、重置和普通按钮.....	78
5.2.4	单选按钮和复选框.....	79
5.2.5	隐藏域	79
5.2.6	多行文本域	79
5.2.7	菜单下拉列表域	79
5.2.8	综合实例	80
5.3	HTML5 新增表单元素	82
5.3.1	<datalist>元素	82
5.3.2	<keygen>元素	83
5.3.3	<output>元素.....	84
5.4	多样的输入类型	85
5.4.1	email	85
5.4.2	url	86
5.4.3	number	87
5.4.4	range	87
5.4.5	date picker	88
5.4.6	color	89
5.5	HTML5 新增的表单属性	91
5.5.1	autocomplete 属性	92
5.5.2	autofocus 属性	93
5.5.3	form 属性.....	94
5.5.4	form overrides 表单重写属性	95
5.5.5	height 和 width 属性	95
5.5.6	list 属性.....	96
5.5.7	min、max 和 step 属性	96
5.5.8	multiple 属性	97
5.5.9	novalidate 属性.....	97
5.5.10	pattern 属性	98
5.5.11	placeholder 属性	98
5.5.12	required 属性	99
5.6	HTML5 表单提交综合实例	100
	本章小结	103
	本章习题	103

第 6 章 CSS3 揭秘	105
6.1 CSS 简介	105
6.2 CSS 规则的组成	107
6.2.1 CSS 注释	108
6.2.2 长度单位	108
6.2.3 颜色单位和 URL 值	109
6.3 在 HTML 文档中放置 CSS 的几种方式	110
6.3.1 内联样式表	110
6.3.2 嵌入一张样式表	110
6.3.3 链接到一张外部的样式表	111
6.4 CSS 普通选择器	111
6.4.1 HTML 选择器	111
6.4.2 类选择器	112
6.4.3 id 选择器	112
6.4.4 关联选择器	113
6.4.5 组合选择器	113
6.4.6 伪元素选择器	113
6.5 CSS 常见的样式属性和值	114
6.5.1 字体属性	114
6.5.2 颜色属性	115
6.5.3 背景属性	115
6.5.4 文本属性	116
6.5.5 边框属性	117
6.5.6 鼠标光标属性	118
6.5.7 列表属性	119
6.5.8 CSS 综合实例	120
6.6 CSS3 概述	122
6.6.1 CSS3 在选择器上的支持	122
6.6.2 CSS3 在样式上的支持	122
6.6.3 CSS3 对于动画的支持	123
6.6.4 在实际开发中该如何使用 CSS3	123
6.7 CSS 复杂选择器	123
6.7.1 基本选择器	123
6.7.2 多元素的组合选择器	124
6.7.3 属性选择器	124
6.7.4 结构性伪类选择器	125



6.8	CSS3 属性	129
6.8.1	使用 CSS3 属性前的准备	130
6.8.2	边框属性	130
6.8.3	背景属性	132
6.8.4	文本属性	133
6.8.5	用户界面属性	133
6.8.6	动画属性	134
6.8.7	多列布局属性	134
6.8.8	渐变属性	135
6.8.9	透明属性	136
6.8.10	旋转属性	136
6.8.11	服务器端字体属性	136
	本章小结	137
	本章习题	137
第 7 章	CSS3 属性特效	140
7.1	新增颜色模式	140
7.2	文字	142
7.2.1	文字阴影	143
7.2.2	文字描边	146
7.2.3	文字排版	150
7.2.4	定义省略文本的处理方式	152
7.3	自定义文字	153
7.4	弹性盒模型	156
7.5	盒模型阴影	164
7.6	倒影	167
7.7	CSS3 分栏布局	173
7.7.1	列个数和列宽度	173
7.7.2	列之间的缝隙间隔宽度	175
7.7.3	分栏间隔符	176
7.8	圆角	177
7.8.1	border-radius 属性	177
7.8.2	单个圆角的设置	180
7.9	边框	181
7.9.1	边框图片 border-image	182
7.9.2	自适应的圆角效果	186

7.10	渐变	188
7.10.1	CSS3 渐变介绍	188
7.10.2	线性渐变	189
7.10.3	线性渐变实例	189
7.10.4	径向渐变	194
7.10.5	径向渐变实例	194
7.11	CSS3 背景	198
7.11.1	多背景	198
7.11.2	background-size	200
7.11.3	background-origin	202
7.11.4	background-clip	203
7.12	遮罩	204
7.13	transition 过渡	206
7.14	2D 变换	207
7.14.1	translate()方法	208
7.14.2	rotate()方法	209
7.14.3	scale()方法	212
7.14.4	skew()方法	213
7.15	3D 变换	215
7.15.1	transform-style	216
7.15.2	perspective 景深	216
7.15.3	perspective-origin 景深基点	218
7.15.4	3D 位移	219
7.15.5	3D 旋转	222
7.15.6	3D 缩放	223
7.15.7	3D 盒子	225
7.16	animation 动画	226
7.16.1	关键帧 keyframes	227
7.16.2	animation 动画属性	229
	本章小结	229
	本章习题	230
第 8 章	DIV+CSS 网页标准化布局	232
8.1	DIV+CSS 页面布局的优势	232
8.2	“无意义”的 HTML 标签<div>和	233
8.3	W3C 盒子模型	234



8.4	和页面布局有关的 CSS 属性	236
8.5	盒子区块框的定位	238
8.5.1	相对定位	238
8.5.2	绝对定位	239
8.6	使用盒子模型的浮动布局	240
8.6.1	设置浮动	240
8.6.2	行框和清理	242
8.7	DIV+CSS 的兼容性问题	244
8.7.1	不同浏览器解释盒子模型的差异	245
8.7.2	遵循 W3C 标准设置浏览器	246
8.8	使用盒子模型设计页面布局	247
8.8.1	居中设计	247
8.8.2	设置两列浮动的布局	248
8.8.3	设置三列浮动的布局	250
8.8.4	设置多列浮动的布局	251
8.9	DIV+CSS 网站首页布局示例	252
8.9.1	HTML 文件的设计	253
8.9.2	CSS 文件的设计	254
	本章小结	256
	本章习题	256
第 9 章	响应式布局	259
9.1	响应式布局的优缺点	259
9.2	如何设计响应式布局	260
9.3	响应式布局实例	261
9.4	Media Queries 模块的使用方法	263
9.4.1	语法结构及用法	264
9.4.2	可用的设备类型	265
9.4.3	可用的设备特性参数	266
9.5	在移动设备上设置原始大小显示	268
9.6	响应式网站的内容设计	268
9.6.1	响应式图片显示内容设计	269
9.6.2	响应式文字显示内容设计	269
9.7	响应式网站的设计流程	270
	本章小结	271
	本章习题	272

第 10 章 认识和使用 Sass	273
10.1 初识 Sass.....	273
10.1.1 Sass 是什么	274
10.1.2 Sass 的作用	274
10.1.3 Sass 的安装	277
10.1.4 Sass 的使用和编译	279
10.2 Sass 基本语法与使用实例	283
10.2.1 Sass 基本语法	283
10.2.2 Sass 使用实例	288
本章小结	292
本章习题	292
第 11 章 栅格布局.....	294
11.1 栅格	294
11.2 盒子模型	296
11.3 栅格实例.....	297
11.4 Bootstrap 框架	298
11.4.1 Bootstrap 现状.....	299
11.4.2 栅格系统	299
本章小结	313
本章习题	313
第 12 章 Bootstrap 的快速入门.....	315
12.1 Bootstrap 的内容概述与整体理解	315
12.1.1 全局 CSS 样式	316
12.1.2 组件	317
12.1.3 JavaScript 插件.....	319
12.2 Bootstrap 搭建环境.....	320
12.3 Bootstrap 全局 CSS 样式.....	321
12.3.1 全局 CSS 样式的栅格系统和响应式布局	322
12.3.2 全局 CSS 样式的表单	324
12.4 Bootstrap 组件.....	324
12.5 Bootstrap 的 JavaScript 插件	326
本章小结	328
本章习题	328



第 13 章 Bootstrap 的实战	330
13.1 实战概述	330
13.2 实战需求	330
13.3 实战准备	333
13.3.1 Sass 配置	334
13.3.2 HTML 的基本模块	335
13.4 顶部工具栏	336
13.4.1 Bootstrap 的字体图标组件	336
13.4.2 Bootstrap 的下拉菜单组件	337
13.4.3 Bootstrap 的输入框组组件	339
13.4.4 Bootstrap 的导航组件	341
13.4.5 顶部工具栏 PC 端内容填充	342
13.4.6 顶部工具栏 PC 端样式优化	345
13.4.7 实战顶部工具栏移动端	352
13.5 页面导航条	353
13.5.1 Bootstrap 的导航条组件	354
13.5.2 实战页面导航条——内容填充	356
13.5.3 实战页面导航条——样式优化	358
13.6 banner 区	360
13.6.1 Bootstrap 的 JavaScript 轮播图插件	360
13.6.2 实战 banner 区	362
13.7 推荐位	364
13.7.1 Bootstrap 的缩略图组件	364
13.7.2 Bootstrap 的面板组件	365
13.7.3 实战第一种风格的推荐位——内容填充	366
13.7.4 实战第一种风格的推荐位——样式优化	369
13.7.5 实战第二种风格的推荐位——内容填充	372
13.7.6 实战第二种风格的推荐位——样式优化	375
13.8 网站的脚部	379
13.9 用户登录	380
13.9.1 Bootstrap 的表单全局 CSS 样式	380
13.9.2 Bootstrap 的模态框 JavaScript 插件	382
13.9.3 实战用户登录	384
本章小结	386
本章习题	387
附录 A	388

第1章

介绍 HTML5



在很多人眼里，HTML5 与互联网营销密切相关，其实从开发者的角度而言，它是一种网页标准，定义了浏览器语言的编写规范。伴随 HTML5 标准尘埃落定，浏览器对 HTML5 特性的逐步支持，再加上国内对 HTML5 大力推广与应用，又出现了各种各样的 HTML5 平台。HTML5 现在已经是一个富含多元化的市场机会，它强大的 Web 应用开发能力让人们不得不转换固有的互联网思维，寻找新的网页解决方案；开始由技术应用转向整个行业的良性生态化，而且还存在巨大的上升空间。现在 HTML5 会由广告行业逐渐向游戏行业、广播电视行业和媒体行业转型，不远的未来还能把触角伸得更远。本章将对 HTML5 进行全面分析，让读者了解 HTML5 是什么、能做什么、学什么、HTML5 和其他技术的关系，以及 HTML5 的行业发展等。希望读者可以读完本章，掌握了宏观印象后会对后面的学习有很大帮助。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

1.1 了解HTML5 的主流应用

HTML5 是 Web 开发世界的一次重大改变，它代表的是未来 Web 开发的趋势。介绍使用 HTML5 和其优势的文献太多了。它看起来很神秘，感觉像喷气背包或飞行汽车，就像被



“神化”了，在 Web 世界只要跟“炫酷”沾上边的都说是 HTML5 开发的，先不管这种说法对不对，我们先来了解目前一些 HTML5 的主流应用，看看 HTML5 的能力。

1.1.1 表单的强大

表单在 Web 页面中很常用，HTML4 中的表单功能单一，例如一个表单只能将数据提交到一个服务器位置，当然使用 JavaScript 也可以实现提交到多个位置，不过代码就很烦琐了。而且表单验证等功能都要在 JavaScript 里面写，当然用前端框架验证可能会很简单，但却避免不了浏览器加载 JavaScript 代码缓慢等问题。所以 HTML4 的表单实际使用时非常依赖 JavaScript，用起来不是很方便。而 HTML5 就可以不用那么依赖 JavaScript，针对种种问题，HTML5 提供的新功能都可以解决。另外，HTML5 还专为移动平台定制了一系列表单元素，只需要使用 HTML5 中新增加的特定标签属性，就可以完成对不同样式键盘的调用，简捷方便，如图 1.1 所示。

图 1.1 HTML5 表单示例

1.1.2 响应式页面布局

网站是由网页组成的，网页则用来呈现内容和用户互动，而内容的摆放不能太随意，在配色、字体及布局排版方面要精心设计，这样才能制作出漂亮的网站，让用户和搜索引擎喜欢。页面的布局也经历了几次发展，从最早的表格排版，到使用 DIV+CSS 标准化网页布局，再到现在使用 HTML5 来实现，每一次变革都是由新技术的使用有着明显的优势所决定的。

除有利于搜索引擎抓取内容外，HTML5 做页面布局的优势还包括易用性、代码清晰和功能强大三个方面。

1. 易用性

使用 HTML5 创建网站更加简单，主要是因为新的 HTML 标签如<header>、<footer>、<nav>、<section>和<aside>等，可以使阅读者更容易访问内容。而在上一代布局技术 DIV+CSS 的使用中，即使定义了 class 或 id，阅读者也无法去了解给出的一个 DIV 究竟是什么。使用新的语义标签可以更好地了解 HTML 文档，并且创建更好的使用体验，如图 1.2 所示。

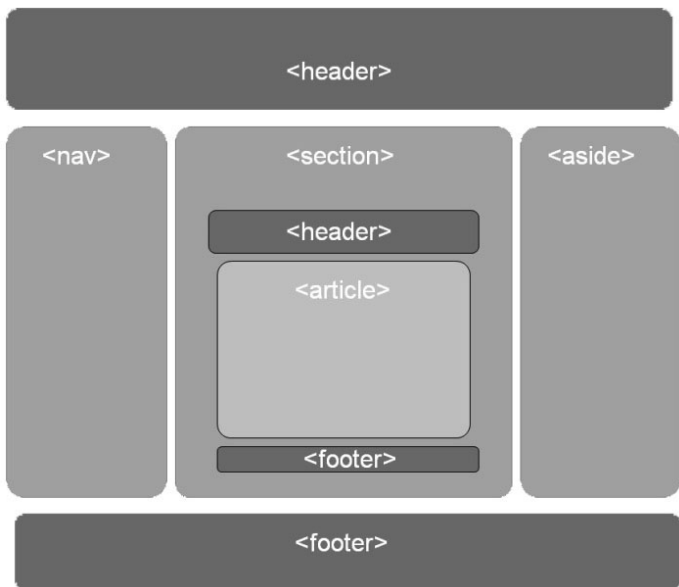


图 1.2 HTML5 中的语义标签布局展示

2. 代码清晰

如果你对简洁、优雅、容易阅读的代码有所偏好，那么 HTML5 绝对是为你量身定做的编程语言。HTML5 可以写出简单清晰且富于描述、符合语义学的代码，便于读者分开样式和内容。而且 DOCTYPE 没有更多内容了，不需要复制粘贴一堆无法理解的代码，也没有多余的 head 标签。除了简单，HTML5 最大的好处在于它能在每个浏览器中正常工作，即使是“声名狼藉”的 IE 6。

3. 功能强大

HTML5 支持响应式页面布局，无论用户使用何种终端访问你的网站，HTML5 都能够自动识别适应终端设备的分辨率及宽度，让你的网站在众多设备中无缝浏览（包括显示器、便携设备、电视机等）。对于智能手机和平板电脑的逐渐普及，普通的网站对于这些持有移动



设备的用户来说，访问无疑是困难的，他们必须在设备上放大或缩小整个网页，以便能够使字体大小适合阅读，但稍不小心就可能会点错进入其他区域，这种状况在响应式网页设计中会有所改善。响应式页面布局就是一个网站能够兼容多个终端，而不是为每个终端做一个特定的版本。这个概念是为解决移动互联网浏览而诞生的，响应式页面布局可以为不同终端的用户提供更加舒适的界面和更好的用户体验，而且随着目前大屏幕移动设备的普及，可以说响应式页面布局是大势所趋，现在越来越多的网站开始采用响应式页面布局方案，如图 1.3 所示。

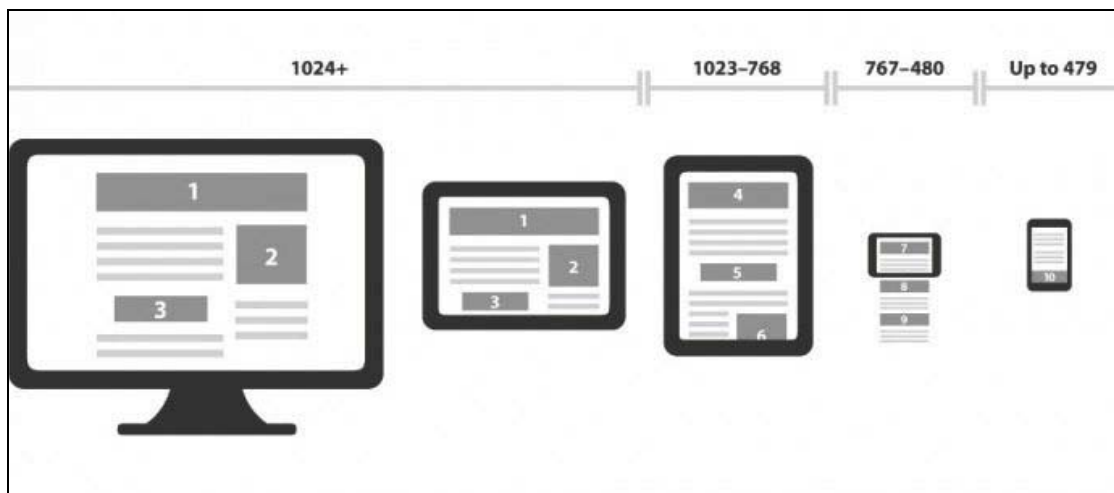


图 1.3 Web 响应式页面布局

1.1.3 与用户交互的特效

十多年前做一个页面，只要结构清晰并且内容呈现完整，就是一个非常不错的网站了。而现在的用户对视觉的体验要求越来越高，在用户的潜意识里，页面做得越炫代表公司实力越强。无论是整体页面风格特效（目前采用类似 PPT 中的幻灯片播放效果居多），还是页面中的局部特效（如炫酷的导航），以及在微信中传播的品牌宣传信息等，都成了 HTML5 开发的主战场。而过去十几年，Web 功能中的特效与互动多数是由 Adobe Flash 主宰的，而现在目光转向了 HTML5。HTML5 的画图标签允许做更多的互动和动画，就好比 we 使用 Flash 达到的效果。它会带来统一的网络，无论是笔记本、台式机，还是智能手机，都应该很方便地浏览基于 HTML5 的网站。因此在设计网站的时候，开发者需要重新考虑用户体验、网站浏览、网站结构等因素，使得这个网站对任何硬件设备都通用。HTML5 同样也拥有很多 API，可以创建更好的用户体验以及更加动态的 Web 应用程序，如图 1.4 所示。

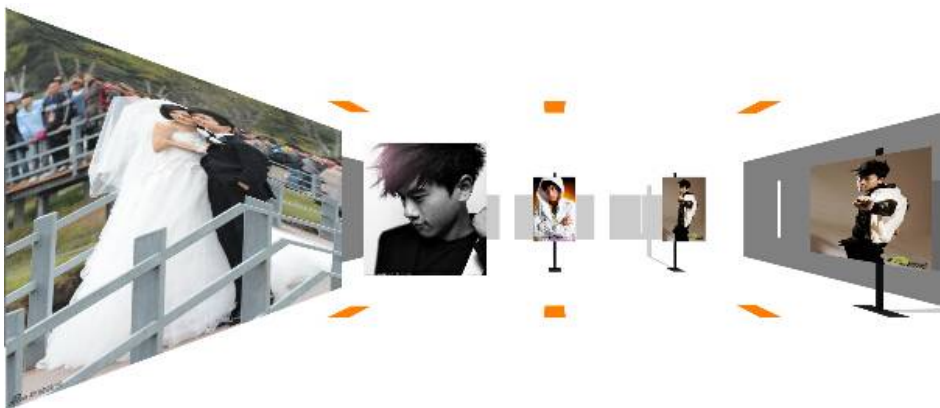


图 1.4 HTML5 开发的页面 3D 特效

笔者提示：无论是 Flash 还是 HTML5，其实并不特别重要，由于技术变化太快，设计和互动才是最关键的。

1.1.4 微网站的制作

智能电话的升级，使用户快速地从 PC 端转向移动端，并随着微信用户的增加及微信公众平台开放平台中的服务号和订阅号的大量应用，微信网站的应用快速增长，诸如微官网、微商城、微活动、微海报、微分销等的开发已经成为主流业务，如图 1.5 所示。



图 1.5 微网站产品展示

HTML5 正在逐渐成熟，在 PC 端应用还有一些不兼容的问题需要解决，但在移动端的浏览器中运行已经非常稳定，可以说 HTML5 的应用是在移动端率先发展起来的。几乎所有人都热衷于开发独立的移动应用，但 HTML5 很可能是独立移动应用的终结者。由于可以将 HTML5 应用的功能直接封装到 APP 中，这很可能引导移动技术潮流重新回到浏览器时代。HTML5 允许开发者在（移动）浏览器内开发应用，所以如果要制定一项桌面或移动应用的长期发展策略，就可能需要考虑这一点。



1.1.5 基于HTML5 的移动APP开发

不同的操作系统中，需要安装用不同技术开发的 APP。移动端的操作系统有很多种，但主要有两大分支，一种是在苹果 iOS 操作系统中使用 Objective-C 语言开发 APP，另一种则是在谷歌 Android 操作系统中使用 Java 开发包。所以同一款 APP，我们需要组织多支开发团队，分别针对不同的操作系统进行开发，并且技术难度要求高、开发周期长，所以开发成本当然也很高。就目前来说，在 HTML5 规范中，已经加入了相机、磁力罗盘、GPS 信息的支持，依托于网络并基于信息流方式及类似方式的应用最适合使用 HTML5 进行开发，比如微博、社交、新闻、商城以及地图、导航等应用类型。如果能用一个统一的 HTML5 来替代 Android 和 iOS 并行开发的双重成本，那么不正是企业目前急需的技术吗？如图 1.6 所示。



图 1.6 可以用 HTML5 开发的 APP 展示

基于 HTML5 开发的 APP 可以降低企业的开发成本，提高开发效率，而且性能更好，耗电量更低，方便升级，打开即可使用最新版本，免去重新下载升级包的麻烦，在使用过程中就直接更新了离线缓存。用户想要什么，HTML5 就能提供给用户什么。

1.1.6 HTML5 游戏

HTML5 提供了一种非常伟大的、移动友好的方式去开发有趣互动的游戏。如果要开发 Flash 游戏，那么相信你会喜欢上 HTML5 的游戏开发。HTML5 游戏通常称作轻游戏，有如下几个明显的特点：

- HTML5 小游戏玩的时间短又不失乐趣，如神经猫、疯狂手指、数钞票、见缝插针等小游戏都属于这类，可以在 1 分钟之内结束。
- HTML5 游戏非常简单，像快餐类游戏，因为现在轻游戏中传播量最大的都是有趣的益智类游戏。
- HTML5 适合开发竞技类游戏，这体现在分数上，也有像超过宇宙百分之多少人之类的，还有本地存储分数等，稍复杂一点还可以加入排行榜。
- HTML5 游戏可以抓热点、拼创意，如疯狂手指的创意来自一个工程师，被另一工程师开发后上线，上线第二天就达到近千万浏览量，创意爆款可遇而不可求。

HTML5 游戏如图 1.7 所示，HTML5 游戏归属于基于 Web 的游戏，传播途径也非常广泛，可以在微信中转发，打开即玩，让用户可以在任何时间、任何地点不用下载就能玩轻游戏。还可以用推送消息的方式做新游戏的推广，微信朋友圈的游戏可以作为一种很好的运营工具来使用。但从技术上看，随着手机和网络的发展，HTML5 开发的轻游戏相比 APP 显然更有优势、更利于用户，开发成本也相对较低，相信它一定能给游戏行业带来很大的变化。



图 1.7 基于 HTML5 开发的游戏

1.1.7 多媒体的应用

在页面中播放媒体一直都是一件非常“可怕”的事情，需要使用<embed>和<object>标签，并且为了使它们能够正确播放，必须赋予一大堆参数，媒体标签将会非常复杂，并且有很多令人迷惑的代码。应用 HTML5 可以让你忘记 Flash 和其他第三方应用，让视频和音频通过 HTML5 新增的媒体标签来访问资源，就像在页面中放上一张图片那样容易，如图 1.8 所示。



图 1.8 应用 HTML5 新标签添加视频播放器

1.2 什么是HTML5

HTML5 开发现在很火爆，它是一门技术，更是一个概念。它让我们的工作模式、交互模式以及对应用和游戏的体验发生了翻天覆地的变化，很多人都知道 HTML5 这门技术，也常把 HTML5 读作 H5（简称）。其实有些人对 HTML5 的认识是存在一些误区的，例如微信上出现一个应用就说是 HTML5 做的、看到炫酷的页面也说是 HTML5 做的，看到一个网页游戏还说是 HTML5 做的，以及看到一个数据提交的小表单也理解成 HTML5 做的。这种理解是片面的，虽然确实有 HTML5 的身影存在，但如果 HTML5 不用 JavaScript，就像汽车没有发动机；如果 HTML5 不用 CSS，就像汽车只有框架而没有绚丽的外观，基于 HTML5 的开发必须要和其他多种技术配合才能实现。当然也不用太“较真儿”，你可以把 HTML5 开发看作代表 Web 开发，或当作前端开发的统称，或者把 HTML5 看作 HTML、CSS3、JavaScript、jQuery 等开发技术的综合代名词，通过本节的学习相信读者能够真正认识 HTML5。

1.2.1 HTML5 和HTML的关系



HTML5 是下一代 HTML 标准，要想了解什么是 HTML5，就要先认识它的祖先 HTML。HTML 全称为超文本标记语言（HyperText Markup Language），是被用来结构化细节、定义文档外观和语义的一种标记语言。说白了，HTML 就是在 Web 世界中，

将内容放到网页中的技术，内容包括文档、链接、图片、视频等，并能做一些简单的格式布局。

早期的 HTML 非常简单，1980 年，为使世界各地的物理学家能够方便地进行合作研究，创建了适用于其系统的 HTML。英国计算机科学家、万维网的发明者 Tim Berners-Lee，设计的 HTML 以纯文字格式为基础，可以使用任何文本编辑器处理，当然最初仅有少量标记 (TAG) 且易于掌握运用。而随着 HTML 使用率的提高，人们开始不满足只能看到文字。1993 年，还是大学生的 Marc Andreessen 在其 Mosaic 浏览器加入标记，从此以后在 Web 页面上就可以浏览图片了。但人们认为仅有文字和图片还不够，从而希望可以将任何形式的媒体都加到网页上，因此 HTML 不断地扩充和发展，经过 20 多年的发展历经了多个版本，如表 1.1 所示。

表 1.1 HTML 20 多年的发展版本

HTML 版本	发布时间	备 注
HTML1.0	1993 年 6 月	作为互联网工程小组 (IETF) 工作草案 (并非标准)
HTML2.0	1995 年 11 月	作为 RFC 1866 发布，在 RFC 2854 于 2000 年 6 月发布之后被宣布已经过时
HTML3.2	1996 年 1 月 14 日	W3C 推荐标准
HTML4.0	1997 年 12 月 18 日	W3C 推荐标准
HTML4.01	1999 年 12 月 24 日	对 HTML4.0 有微小改进，W3C 推荐标准
XHTML1.0	2000 年 1 月 26 日	是 W3C 推荐标准，后来经过修订于 2002 年 8 月 1 日重新发布
XHTML1.1	2001 年 5 月 31 日	W3C 推荐标准
XHTML2.0	W3C 工作草案	XHTML2.0 因为改动过大、学习这种新技术的成本过高而最终导致胎死腹中。现在常用的还是 XHTML1.0 的标准
HTML5	2014 年 10 月	由 W3C 完成标准制定，是 HTML 最新的修订版本

HTML5，第 5 版超文本标记语言，单纯从技术的角度来看，HTML5 就是 HTML 标准



的最新版本，于 2014 年 10 月由万维网联盟（W3C）发布为正式推荐标准。它是 HTML 自 1991 年问世以来，最具变革价值的技术规范，历经多年修订与完善才制定完成。HTML5 又不仅仅是 HTML4.0 的下一个版本，因为它同样支持 HTML4.0 之后的网页规范，是首个将 Web 作为应用开发平台的 HTML 标准。而从其他角度来讲，HTML5 现在如此火爆的原因总结如下：

(1) HTML5 增加了许多新特性，使网页能力变得更强，这让许多以前不切实际的想法变成了现实，也让很多难以实现的功能变得很简单。

(2) 近几年移动互联网越来越普及，跨平台、跨终端的需求越来越明显，这也为 HTML5 的发展提供了契机。

(3) 对于应用或游戏开发来说，一套跨平台的标准更易于节约开发成本，让开发工作从繁重的多平台版本中解脱出来。

所以 HTML5 很像当年 Web 2.0 的概念，基于现有的技术，让用户体验到不一样的互联网世界。

1.2.2 HTML和CSS的关系



HTML 是描述网页的标记语言，将内容放到网页上，虽然 HTML 本身也自带一些样式功能，通过自身的属性来实现一些特定的效果，但制作出来的只能是一个不美观的网页。最主要的是在 HTML 里面有一些标签有一定的语义化，有些标签和属性在不同浏览器中的兼容性并不一样，在标签里面添加很多属性，造成文档内容复杂，使得独立于外观（表现层）的网站开发起来越发困难。为了解决上述难题，W3C 在 HTML4.0 的基础上研究出了样式，也就是 CSS。所以现在可以很清楚地知道它们之间的关系：

(1) HTML 定义网页的结构，主要让页面的内容结构化、块状化。

(2) CSS 控制 HTML 的标签，定义所需要的样式，使得结构和样式独立开来。

这样使得样式和结构分离，达到了我们的初衷，内容更加清晰可读。CSS 给我们的站点开发带来了很多好处，突出的优点包括：

(1) CSS 使得我们的内容更加清晰，代码可读性更高。主要原因是不用在标签里面写大量属性和重复性的代码。

(2) 提高了网页的浏览速度，减少了冗余的代码。在 HTML 里面，原本有很多结构块，效果展示一致，只是内容变化。在 HTML 早期，需要对每一块设置相同的属性，造成代码大量冗余，文件的大小自然也有所增加，对页面的浏览速度有所减缓。

(3) 实现结构和样式的分离。HTML 标签和 CSS 样式的独立，使得网站页面在开发和

改版的时候，容易且简单。CSS 没有出现独立成为外部文件之前，基本上页面改版就等于重做，或者只是局部改动，根本不敢改，主要也是因为修改起来很困难，所以开发会相对困难很多。CSS 的出现，实现了最初的宗旨——“结构和样式分离”。

需要注意的是，CSS 始终控制标签，所以 CSS 离不开 HTML，HTML 没有 CSS 也只能有基本的效果。但自从有了 CSS，控制出的页面更加美观，代码更加清晰可读，改版更容易，兼容性也更好了。所以 HTML 和 CSS 相辅相成，HTML 控制结构，为整个网页搭建框架；CSS 控制样式，为整个页面“穿衣服”。

1.2.3 HTML5 和 CSS3 的关系



HTML5 是第 5 版 HTML 的标准，CSS3 则是第 3 版 CSS，新增了一些非常实用的选择器和样式属性，并且 CSS3 语言开发是朝着模块化发展的。以前的规范作为一个模块实在是太过于庞大且比较复杂，所以把它分解为一些小的模块，更多新的模块也被加入进来。这些模块包括盒子模型、列表模块、超链接方式、语言模块、背景和边框、文字特效、多栏

布局等。HTML 和 CSS 结合开发使用非常紧密，HTML5 和 CSS3 的结合使用还要更近一些，正在逐渐替代上一代产品。HTML5 的一些高级特效，如圆角、阴影、响应式页面布局、各种字体加载等效果，必须有 CSS3 的配合才能实现得更完美。所以有用到 HTML5 技术的地方，就离不开 CSS3，但在概念上两者的名称很少一起出现，因为名称太长，我们一般把目光放在 HTML5 上面。所以在谈论 HTML5 技术时，其概念中也包含了 CSS3 技术。

1.2.4 HTML5 和 JavaScript 的关系



JavaScript 是实现 HTML5 的重要语言。长久以来，JavaScript 一直都是在 HTML 中实现动态效果的不二之选，而 JavaScript 在一些程序员眼里是编程语言中的“二等公民”。早先，它经常是很多安全问题的源头，就像胶水一样，它能把 HTML 应用与样式粘到一起，但没有人拿它来正正规规地编写程序，这样的情形太普遍了，而 Java、PHP、C 语言等才是真正能用来编写程序的语言。而过去几年间，随着 Web 的发展，

程序员对 JavaScript 的态度有了质的改变，认为 JavaScript 已经“长大成人”，但其实 JavaScript 一直都是一种十分强大、成熟、深得人心的语言，JavaScript 自身就具备很强的表达能力，还有众多的库和开发工具。而且随着 jQuery、JSON、Node.js 和 HTML5 的出现，



JavaScript 就更加完善了，其逐渐成为每个开发人员都知道的编程语言。如果你要学一门新的编程语言，那么非 JavaScript 莫属。

其实 HTML5 本身就是使用一些标签而已，我们谈论的 HTML5 或前端技术，其实说的就是 JavaScript。例如做一个小游戏，用 HTML5 的几个标签可能只有几十条代码，而整个游戏要几千条 JavaScript 代码，但这个游戏会被叫作“HTML5 开发的小游戏”，其实这些完全归功于 JavaScript。由此可见，HTML5 并不是以尖括号为特征的标签语言的一次大的改进，其实质是赋予了 JavaScript 更强大的能力。另外，诸如 WebGL 库支持在 HTML5 的画布中绘制实时的 3D 图形、HTML5 的地理位置支持在浏览器中实现 LBS（Location Based Service）应用，这些现在都是手机的基本配置。而持久存储及离线功能，则为开发能与桌面应用相媲美的，但却在浏览器中运行的全功能应用奠定了基础。目前，就连增加多点触摸事件的实验性的库也已经出现了。这一切，无一不是实实在在的 JavaScript 特性，HTML5 只是为这些高级功能的发挥提供了平台。

1.3 HTML5 的靠山

无规矩不成方圆，软件开发当然也不例外。Web 开发涉及的厂商和技术非常多，所以必须要有参考的标准，而且需要一系列的标准。Web 程序都是通过浏览器来解析执行的，通过页面的展示内容与用户互动，所以 Web 标准不仅要求各个浏览器要遵循，开发者也要遵循相同的标准。但和 Web 标准相关的制作组织机构很多，如 W3C、IETF、ECMA 和 WHATWG 等，哪些是我们需要了解的以及我们需要掌握什么信息，都将在本节进行详细介绍。

1.3.1 W3C是什么



W3C 创建于 1994 年，是万维网（World Wide Web）联盟的缩写。W3C 是制定网络标准的一个非营利性组织，是 Web 技术领域最具权威和影响力的国际中立性技术标准机构，如 HTML、CSS、XML 的标准就是由 W3C 来制定的。

W3C 致力于对 Web 进行标准化，创建并维护了 WWW 标准。

W3C 为解决 Web 应用中不同平台、技术和开发者带来的不兼容问题，保障 Web 信息的完整流通，制定了一系列标准并督促 Web 应用开发者和内容提供者遵循这些标准。标准的内容包括使用语言的规范，开发中使用的导则和解释引擎的行为，等等。但是，W3C 制定的 Web 标准似乎并非强制，而只是推荐标准（每项 W3C 推荐的发展是通过由会员和受邀专家组成的工作组来完成的。工作组的经费来自公司和其他组织，并会创建一个工作草案，最后是一

份提议推荐。一般来说,为了获得正式批准,推荐都会被提交给 W3C 会员和主任)。因此部分网站仍然不能完全实现这些标准,特别是使用早期所见即所得网页编辑软件设计的网页,往往会包含大量非标准代码。到目前为止,W3C 已发布了 200 多项影响深远的 Web 技术标准及实施指南,有效促进了 Web 技术的互相兼容,对互联网技术的发展和應用起到了基础性和根本性的支撑作用。W3C 之所以被业界所推崇,离不开其创始人、被业界公认为互联网之父的蒂姆·伯纳斯·李。如果蒂姆·伯纳斯·李当初为自己发明的“WWW”申请专利,那么他现在可以在财富上与比尔·盖茨一较高低,但他选择了无偿向全世界开放,让所有人都有机会接触到互联网。

1.3.2 IETF是什么



国际互联网工程任务组(The Internet Engineering Task Force, IETF)是一个公开性质的大型民间国际团体,由为互联网技术工程及发展做出贡献的专家自发参与和管理,汇集了与互联网架构和互联网顺利运作相关的网络设计者、运营者、投资人和研究人员,并欢迎所有对此行业感兴趣的人士参与,任何人都可以注册参加 IETF 的会议。IETF 成立于 1985 年年底,是全球互联网最具权威的技术标准化组织,其主要任务是负责互联网相关技术规范的研发和制定,当前绝大多数国际互联网技术标准都出自 IETF。

IETF 机构与 W3C 是互联网行业内两大标准组织。IETF 具有比 W3C 更广泛的责任范围,它负责定义并管理因特网技术的所有方面,包括用于数据传输的 IP 协议、让域名与 IP 地址匹配的域名系统(DNS)、用于发送邮件的简单邮件传输协议(SMTP)等。当前 IETF 正在推动的两大标准是互联网协议 IPv6 和增加一个加密层的 DNSSEC。W3C 主要关注的是 Web 方面,其早期定义了 WWW 的 HTML、CSS、HTTP 和 URL 等基础技术标准,以及现在的 HTML5、CSS3、Web APP 等标准。

W3C 发布的是工作草案和建议,试图为今后通过指定的 Web 协议。IETF 发布的称为征求意见稿和推荐标准,在现实世界中已经使用。在 W3C 创立之时,它的权力与 IETF 重合了,IETF 组织最终将负责 HTML 的权力移交给了 W3C,因为主要的厂商,如 Microsoft 和 Netscape 公司倾向于通过 W3C 工作。HTML 的第一个官方版本是由 IETF 推出的 HTML2.0。后来,W3C 取代 IETF 的角色,成为 HTML 标准制定的组织。从此 IETF 和 W3C 一直保持着合作关系,在许多问题上立场一致。2009 年,互联网协会 ISOC(IETF 的上级机构)宣布为推动作为创建开放网络标准组织的 W3C 的发展,对其进行捐助。ISOC 和 W3C 多年来在许多领域共同努力,并已深刻共享关于互联网发展的价值观。ISOC 的支持将使 W3C 能够发



展其组织结构，以确保继续巩固与世界各地越来越多的开发者和用户的坚实的合作关系。这两个组织将继续独立运作，并保持其长期而非正式的合作。

1.3.3 RFC是什么



RFC 文档也称请求注解文档 (Requests for Comments, RFC), 这是用于发布 Internet 标准和 Internet 其他正式出版物的一种网络文件或工作报告, 内容和 Internet 相关。草案讨论了计算机通信的方方面面, 重点在网络协议、过程、程序, 以及一些会议注解、意见、风格方面的概念。一个 RFC 文件在成为官方标准前一般至少要经历三个阶段: 建议标准、草案标准、因特网标准。在 Internet 上, 任何一个用户都可以对 Internet 某一领域的问题提出自己的解决方案或规范, 作为 Internet 草案提交给 Internet 工程任务组 (IETF)。草案存放在美国、欧洲和亚太地区的工作文件站点上, 供世界多国自愿参加的 IETF 成员进行讨论、测试和审查。最后, 由 Internet 工程指导组确定该草案是否能成为 Internet 的标准。RFC 文档必须被分配 RFC 编号后才能在网络上发布。例如, RFC2026 的内容是 “Internet 标准进程-修订版 3”, RFC1543 的内容为 “RFC 作者指导”, 等等。

1.3.4 WHATWG是什么



WHATWG (Web Hypertext Application Technology Working Group) 即网页超文本应用技术工作小组, 是一个以推动网络 HTML5 标准为目的而成立的组织。2004 年, WHATWG 由 Opera、Mozilla 基金会和苹果这些浏览器厂商组成。WHATWG 成立的原因是 W3C 意图放弃 HTML, 而力图发展 XML (可扩展标记语言下的一个子集) 技术。WHATWG 邮件列表公布于 2004 年 6 月 4 日, 否决了由 W3C 成员在 W3C 工作室的 Web 标准, 组织中的浏览器厂商建议 W3C 跟随 WHATWG 的 HTML5, 将新的 HTML (标准通用标记语言下的一个应用) 命名为 HTML5, 后来 W3C 采纳了他们的建议。

1.3.5 Web的新标准

HTML5 于 2014 年 10 月由 W3C 发布为正式推荐标准。其实 HTML4.0 和 XHTML 都只是页面文档标记性语言, 使用标记来描述文档。但 HTML5 却具有构建浏览器应用的能力,

重新定义了 Web 开发。在 HTML5 没有应用之前，Web 开发面临两种困境，一是不少人质疑 Flash 的安全性等问题，但却找不到替代它的合适插件；二是程序员总会抱怨 PC 端和移动端应用的多次开发，需要为苹果、安卓、微软等系统设计不同的方案。而 HTML5 提供了良好的解决方案。与 JavaScript、CSS 等紧密结合后，HTML5 一改“网页即文档”的传统局面，大大增强了网页的富媒体特性。以后浏览器不需要类似 Flash 的插件也能实现复杂交互效果，HTML5 的跨平台可用性更令应用的一次开发成为可能。因此，HTML5 的兴起具有非常深远的意义，它已经从简单的标记语言化身为 Web 应用开发的先驱，成为 Web 应用开发的新标准。总的来说，HTML5 官方规范具有以下 4 种核心特性：

(1) 新的语义标记，有利于搜索引擎或辅助技术对页面的理解，加强页面的可访问性。

(2) 新的表单元素，提供文本、数值、日期、时间、颜色等新的输入类型，并引入一些通用属性，允许对表单字段进行调整。

(3) 添加视频和音频等多媒体元素，使网页不需要第三方插件就能实现各种富媒体功能。

(4) 引入 Canvas 元素，能用 JavaScript 在画布上进行绘制，同时支持 2D 和 3D 画图。

由此，HTML5 的技术实现并非完全依赖 HTML5 标准，它有时仍需结合 JavaScript 等。所以一般提及 HTML5 的 Web 开发技术，除 WHATWG 和 W3C 官方定义的标准外，还包括第 3 版层叠样式表 (CSS3)、地理定位 (Geolocation) 等规范。

1.4 HTML5 的曲折发展过程

十年磨一剑，正如我们所看到的那样，HTML5 大潮正来势汹汹。但也正如我们所知道的一样，HTML5 是一种技术标准，它的语义之美、人性之美、简单之美、实用之美，等等，如同一场革命，它的主要应用场景是浏览器，不过由于浏览器引擎的不断进化，HTML5 已经可以和其他应用及技术进行混合并无缝嵌入其中，这让 HTML5 能应用在更广泛的场景中。将 Web 从内容平台推向标准化的应用平台，并一统各个平台阵营的标准。正所谓“天将降大任于斯人也”，HTML5 也同样经历过坎坷的蜕变，同时也在发展过程中留下了一些遗憾的“后遗症”。

1.4.1 HTML5 的诞生

前面不止一次提到 HTML5 是下一代产品，HTML5 是 Web 发展的产物，HTML 的历史可以追溯到很久以前。1993 年，HTML 首次以因特网草案的形式发布。20 世纪 90 年代的人们见证了 HTML 的大幅发展，从 2.0 版到 3.2 版、4.0 版，再到 1999 年的 4.01 版。随着 HTML



的发展，W3C 掌握了对 HTML 规范的控制权。从发展的角度来讲，实际上发布 HTML4.0 之后出现了第一个拐点，1998 年 W3C 便开始着手另一个基于 HTML 的标记语言 XHTML，在 HTML4.01 之后的第一个修订版本 XHTML1.0 颇受争议。这种语言的解析模型十分严格，一个小错误就会使浏览器难以识别，导致页面加载失败。2000 年 1 月 26 日，W3C 一意孤行将 XHTML1.0 作为推荐标准，并表示不会继续发展 HTML，未来的工作会集中在 XHTML2.0 上，意图实现 HTML 向 XML 的过渡。因此，W3C 闭门造车引起了一些巨头的的不满。2004 年，Opera、Mozilla 基金会和苹果这些浏览器厂商成立了一个以推动网络 HTML5 标准为目的的组织 WHATWG，致力于完善 HTML 标准。WHATWG 通过论坛讨论进行多人决策，推出一系列优势明显的 HTML5 规范文档，与 W3C 分庭抗礼。2006 年 10 月，互联网之父即 W3C 创始人蒂姆·伯纳斯·李发表了一篇文章，表示从 HTML 走向 XML 的路是行不通的。直到 2007 年，由于 XHTML2.0 方面的工作陷入了似乎无休止的争论中，W3C 投票后宣布将从 2009 年年底起终止同 XHTML2.0 工作组的合约，转而推动 HTML5 的进展。这样就促使 W3C 于 2007 年放弃 XHTML，转而对 HTML5 进行标准化。HTML5 的发展历程如图 1.9 所示。相比 HTML4.0，HTML5 有了很大变化，它以健壮性为原则，结合 HTML 和 XML 的各种规范，并摒弃 XHTML 过于严格的语法，具备较好的浏览器向后兼容性。“Web 2.0”这个新词正是在那个时候被发明的。Web 2.0 实至名归，开创了 Web 的第二个时代。旧的静态网站逐渐让位于需要更多特性的动态网站和社交网站，这其中的新功能数不胜数。

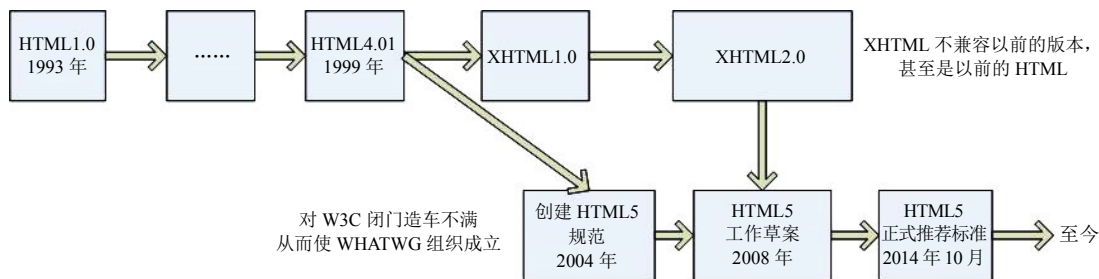


图 1.9 HTML5 的发展历程

1.4.2 浏览器之间的大战



播放电影和音乐要使用播放器，浏览网页则需要使用浏览器。浏览器虽然只是一个设备，就像演出者的舞台，并不是开发语言，但在 Web 开发中必不可少，因为浏览器要去解析 HTML5、CSS3 和 JavaScript 等语言用于显示网页，所以学习 Web 开发一定要先对目前正在使用的浏览器有所了解。由于存在不同的浏览器，浏览器厂商之间还存在着竞争，部分浏览器对个别功能遵循

的标准不一致，从而导致同一段代码在不同的浏览器中会有不一样的解释，显示给用户不一样的结果。市场上常用的客户端浏览器如表 1.2 所示，以后我们还会看到更多浏览器出现。到目前为止，在浏览器市场，谷歌浏览器占据了半壁江山。

表 1.2 常用的客户端浏览器

	Internet Explorer
	微软的 Internet Explorer (IE) 是流行的因特网浏览器之一。它发布于 1995 年，并于 1998 年在使用人数上超过了 Netscape，是 Windows 操作系统中默认的浏览器，现在有多款不同版本的产品
	Netscape
	Netscape 是首个商业化的因特网浏览器，它发布于 1994 年。在 IE 的竞争下，Netscape 逐渐丧失了其市场份额
	Mozilla
	Mozilla 项目是在 Netscape 的基础上发展起来的，是 Linux 操作系统中默认的浏览器
	Firefox
	Firefox 是由 Mozilla 发展而来的新式浏览器，发布于 2004 年，是 Linux 操作系统中常见的浏览器
	Safari
	Safari 是世界上最快、最便于操作的网页浏览器。Safari 具有简洁的外观、雅致的用户界面，是苹果操作系统中默认的浏览器
	Opera
	Opera 是挪威人发明的因特网浏览器。它以快速小巧、符合工业标准、适用于多种操作系统等特性而闻名于世。对于一系列小型设备，诸如移动电话和掌上电脑来说，Opera 无疑是首选的浏览器
	Chrome
	Chrome 又称 Google 浏览器，是一个由 Google (谷歌) 公司开发的网页浏览器。该浏览器基于其他开源软件所撰写，包括 WebKit，目标是提升稳定性、速度和安全性，并创造出简单且有效率的使用者界面

浏览器大战的主因是市场份额的竞争。对普通人来说，浏览器就是桌面上那个蓝色的“e”，多年来微软的 IE 浏览器几乎成了浏览器的代名词，借助其操作系统的捆绑，其市场份额自从推出后一直是市场上的“老大”。但是近些年 Firefox 的蚕食和 Chrome 的鲸吞使得其市场份额不断下降，目前世界上流行的浏览器有 Chrome、IE、Firefox 和 Opera。另外，在国内无论是腾讯、UC，还是海豚浏览器，都已经通过对 HTML5 的大力支持来为自己的未来抢占一席之地。事实上，Web APP 的崛起很有可能伴随着新一轮浏览器格局争夺战的进程。众多厂商在手机浏览器领域借着 HTML5 争相发力，HTML5 技术借助其多媒体、跨平台的优势，未来多屏融合的时代将带来巨大的想象空间，一场围绕 HTML5 的浏览器大战已经开



始了。因为 HTML5 能解决非常实际的问题，所以在规范还未定稿的情况下，各大浏览器厂家就已经按捺不住了，开始对旗下产品进行升级以支持 HTML5 的新功能。这样，得益于浏览器的实验性反馈，HTML5 规范也得到了持续完善，HTML5 以这种方式迅速融入对 Web 平台的实质性改进中。

在浏览器发展的历史中，曾经进行过三次划时代的世界大战，而当前正在发生的正是第三次基于 HTML5 技术的争夺。随着 HTML5 标准的发布，其坚持“开放式互联网”的精神引爆了浏览器如今的战争，各大巨头纷纷站出来宣布拥抱 HTML5 标准。而谁能在这一次新标准的赛跑中胜出，谁就可能在接下来的竞争中占据先机。虽然浏览器之间的战争会推进 HTML5 的发展，但在战争中由于 HTML5 标准迟迟没有敲定，各大浏览器厂商各成一派，对 HTML5 一些功能支持出现了很多分支，从而导致 HTML5 开发人员在处理浏览器之间的兼容性上大费周折，也影响了 HTML5 在 PC 端发展的步伐，使得部分开发人员由于不能很好地解决兼容性问题，从而放弃在 PC 端使用 HTML5 技术，继续使用传统方式开发网站。而移动端的浏览器对 HTML5 的支持还都比较好，所以 HTML5 率先在移动端发展起来。

1.4.3 HTML5 技术的应用现状

HTML5 的优良特性很快被各种类型的网站利用，比如文件拖曳到网页的上传功能，多数使用 HTML5 提供的新属性就可以完成，来实现素材的免插件拖放。因此，HTML5 技术实际上在国内已经获得了较广泛的应用与支持。从硬件角度来看，国内手机和平板电脑两种移动设备应用最广，PC 端次之，紧接着是电视和游戏设备。从软件角度来看，桌面浏览器对 HTML5 的支持高于移动浏览器，最高可达 95%；而从整体上而言，移动浏览器对 HTML5 的支持却优于桌面浏览器。根据百度流量研究院统计，2016 年国内桌面浏览器市场份额最大的是 Chrome，约占 39.4%；IE 次之，约占 29.24%。其中，占比高达 17% 的 IE 8.0 对 HTML5 并不友好。然而，微软已宣布停止对 IE 6.0~IE 10.0 的技术支持，并打算放弃 IE 浏览器及 Windows 10 以下的操作系统。因此，低版本 IE 份额正呈快速下降趋势，2016 年降到了 2% 以下。

1.4.4 HTML5 平台的兴起

2014 年，在微信平台的帮助下，HTML5 社交小游戏获得爆炸式传播，同期为 HTML5 平台以提供制作工具服务进入市场的起点。到了 2015 年，越来越多的公司在 HTML5 品牌推广上进行布局。在商业需求的驱动下，HTML5 页面设计的目的性更强，获得最好传播效果的基本是经过一定时间策划，在团队操作下有针对性地进行投放的企业案例。相应地，原有 HTML5 平台也进行了大面积升级。从平台性质而言，HTML5 平台可分为轻营销模板类、

功能引擎类和基础工具类 3 种。

(1) **轻营销模板类**：提供类似 PPT 页面切换的 HTML5 制作工具，通常面向个人用户，部分为企业用户。该类平台的数量较大，只适用于轻度营销，所能提供的页面动态效果局限于翻页，如图 1.10 所示。



图 1.10 易企秀桌面工具编辑界面和作品

(2) **功能引擎类**：提供 HTML5 网页的开发引擎，相当于一套可编辑的源码系统，能让开发者在现成框架内快速创建项目。功能引擎类 HTML5 平台提供游戏和非游戏两种引擎，应用较广的是游戏引擎，通常面向企业用户。该类平台主要提供基于 Canvas（画布）引擎，适用于轻游戏的开发，依赖于开发者，如图 1.11 所示。



图 1.11 Egret Wing 软件设计师视图界面和作品



(3) **基础工具类**：提供用于页面交互的 HTML5 可视化编辑工具，提供底层交互型产品，开发目的、设计原理和实现思路都以交互为基础，主要面向企业用户和部分个人用户，如图 1.12 所示。



图 1.12 HTML5 桌面工具编辑界面和作品

1.4.5 HTML5 行业的发展预测

现在的互联网市场上，HTML5 在快速成长，未来几年将会有很多商业企业或者公司进军 HTML5 领域，HTML5 也会像传统的 Flex、Flash、Silverlight 和 Objective-C 那样，更容易出现在任何一个设备中，形成一套自己独有的生态系统。对于年轻一代的开发者，HTML5 会成为他们的首选技能，HTML5 会形成很大的市场，将会有很多公司需要这方面的人才。到目前为止，越来越多的行业巨头正不断向 HTML5 示好。除了苹果、微软、黑莓，谷歌的 YouTube 已部分使用 HTML5，Chrome 浏览器宣布全面支持 HTML5，Facebook 则不遗余力地为 HTML5 进行着病毒式传播。HTML5 代表了移动互联网发展的趋势，总有一天它将成为主流技术。所以 HTML5 作为一个前端的编程语言，其发展前景是非常可观的。我们可以从 HTML5 的发展方向中看到它的发展前景。

(1) 手机页游的 3D 化是大势所趋。随着硬件能力的提升、WebGL 标准化的普及以及手机页游的逐渐成熟，大量开发者需要创作更加精彩的 3D 内容。

(2) HTML5 移动营销出现更多新玩法。游戏化、场景化、跨屏互动，HTML5 技术满足了广告商对移动营销的大部分需求，从形式到功用，再到传播。

(3) HTML5 技术的成熟，将带来动漫产业的升级。动漫元素本身可通过 HTML5 来强

化创意，动漫形式将具有富媒体的高度交互、MV 影音功能，为读者提供更加场景化的阅读体验。

(4) HTML5 开发移动应用更灵活。采用 HTML5 技术的轻应用、Web APP 以其开发成本低、周期短、易推广等优势，将迅速普及。

(5) 移动视频、在线直播引领视频升级。HTML5 技术将会革新视频数据的传输方式，让视频直播更加高清流畅。而且，视频还将与网页真正融为一体，让用户看视频如浏览动图一般简单轻松。

(6) 资源复用、影游互动、Web VR 让虚拟现实从贵族走向大众化，以及微信很有可能会推出 HTML5 应用市场。

HTML5 取代 Flash 已经是不可逆转的趋势，而伴随着虚拟现实（VR）、增强现实（AR）等高新技术的兴起，HTML5 的未来更增添了不少机遇，可以预见的是，它会比 Flash 走得更远。

1.5 HTML5 的学习线路图

HTML5 现在可以说是前端所有技术的代名词，学习 HTML5 技术并不是简单地学会几个新增的标签，需要学习的语言和工具很多，刚接触它们的学习者会感觉很乱。另外，前端开发也会细分很多个开发岗位，不同的岗位所涉及的技术也会有所差别，所以首先要确定自己的发展定位，收集要学习的内容，整理好学习的顺序。很多时候，成功除了勇气、坚持不懈，更需要方向。也许有了一个好的方向，成功来得比想象中更快。如果在错误的路上奔跑，那么再怎么努力也是白搭。学习 Web 前端技术也是如此，首先应该选择一条正确的学习路线。HTML5 的学习线路图如图 1.13 所示。

如图 1.13 所示，如果需要掌握全部前端技术，则可以分为三个阶段学习，循序渐进地逐步储备自己的知识量。当然也并不需要学习完全部内容才能参加工作，每个阶段都有自己的定位，有对应的工作岗位，只是学习的内容越全面，发展空间也就越广泛。本书覆盖第一阶段全部内容，第二阶段和第三阶段可以参考本书的配套书籍。

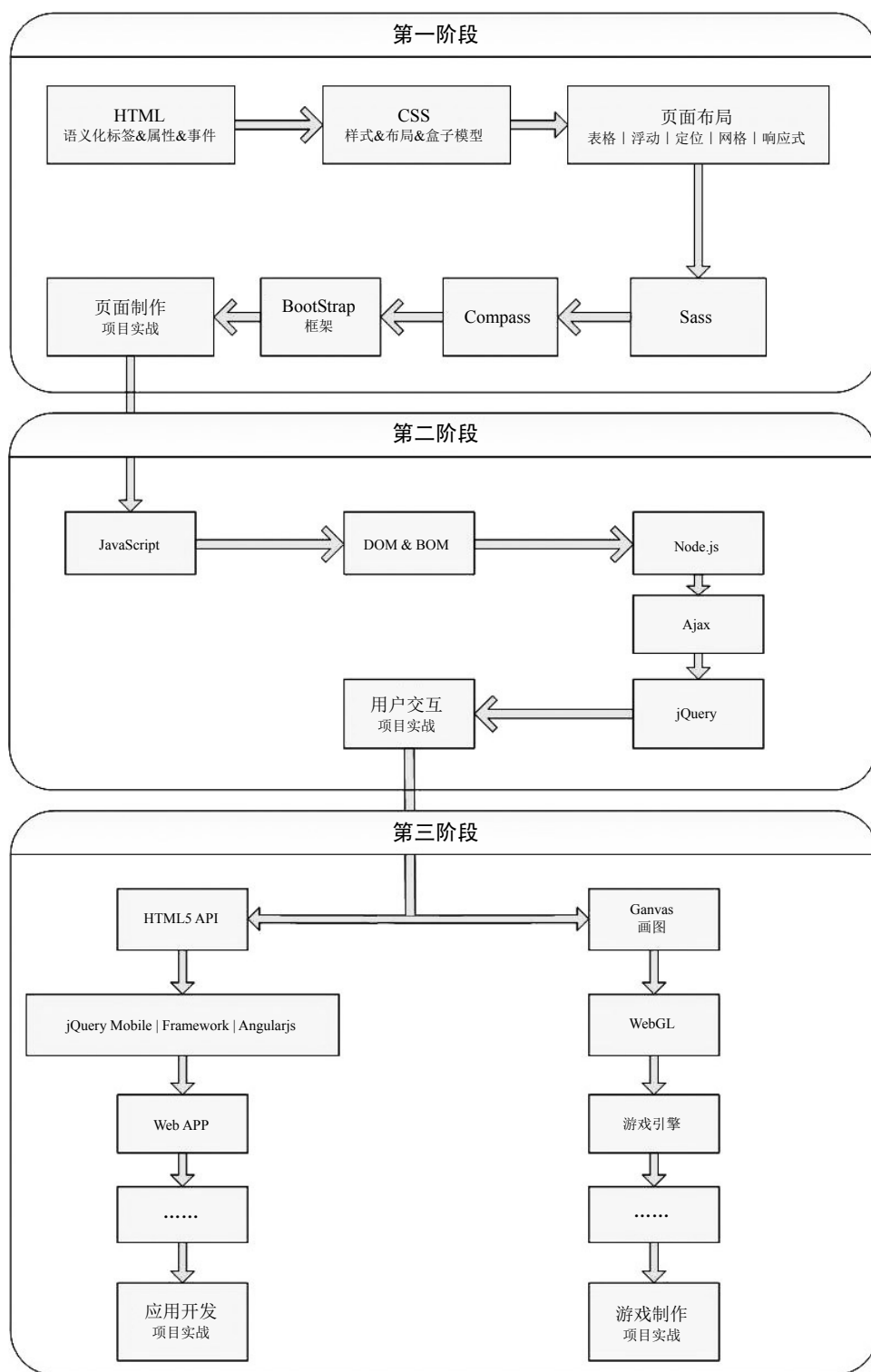


图 1.13 HTML5 的学习线路图

1.5.1 第一阶段——学习网页制作

第一阶段的学习也是 Web 前端技术的开始,虽然对于老手来说这部分技术很简单,但对于新手来讲开头是最难的。技术种类繁多,不知道从何处下手,所以必须有一个良好的学习路径。对于本书学习路径的规划,可以参考以下几个步骤:

(1) 学习前端技术从 HTML 开始,同时也要了解一些 UI 方面的知识。HTML 是为了将内容放到网页上,如文字、图片和表单等,是一个网页的骨架。无论是静态网页还是动态网页,最终返回到浏览器端的都是 HTML 代码,浏览器将 HTML 代码解释渲染后呈现给用户。因此,我们必须掌握 HTML 的基本结构和常用标签及属性。也可以通过 HTML 中的标签和属性做一些简单的页面布局和样式处理,但这是 CSS 的强项,所以这方面的内容仅作为了解即可。目前学习 HTML 可直接从 HTML5 入手,重点掌握最新一代 HTML 语言的语义化标签。HTML 的学习是一个记忆和理解的过程,需要通过代码和效果进行对照学习的方法,来弥补单纯识记 HTML 标签和属性的枯燥乏味。

(2) 学习 HTML 之后,只是掌握了各种“原材料”的制作方法,如果想盖一幢楼房,那么还需要把这些“原材料”按照我们设计的方案组合布局在一起并进行一些样式的美化,这就需要开始学习 CSS 技术。CSS 即层叠样式表,是专门为页面中的 HTML 标签添加样式的语言,是能够真正做到网页表现与内容分离的一种样式设计语言。相对于传统 HTML 的表现而言,其样式是可以复用的,从而极大地提高了我们开发的速度,降低了维护的成本。当然学习 CSS 和学习 HTML 一样,也是从基本的使用语法开始学起,然后掌握 CSS 和 HTML 配合使用的几种方式、CSS 的样式选择器和常用的 CSS 样式属性。也可以从 CSS3 直接开始学习,重点掌握 CSS3 中新增加的样式选择器和新的样式属性。由于样式属性比较多,所以划分为与外观相关的、与布局相关的,以及做盒子模型有关的几部分属性去学习。

(3) 其实学完 HTML 和 CSS 两门技术就可以编写页面了,但只能说你对 HTML 和 CSS 技术有了一定的了解,对其中一些常用的知识还没有完全掌握,也不能在项目中熟练地应用。要想达到技术熟练使用的地步,就必须要学会 HTML 和 CSS 的结合使用,完成各种需求下的页面布局。例如,CSS 中的盒子模型、相对布局、绝对布局等,能够实现对网页中各对象的位置排版进行像素级的精确控制。当然,和语言发展及应用场景不同,CSS 有很多种页面布局方法,如表格布局、浮动布局、定位布局、网格布局,以及响应式页面布局等。

(4) 虽然能布局页面就已经算是一名合格的页面制作人员了,但由于 CSS3 为了能兼容各种浏览器有很多样式属性,同一个属性要不同的前缀写多次才能做到,特别是大型项目中 CSS 属性非常多。SASS 是世界上最成熟、稳定、功能强大的专业级 CSS 扩展语言,提供了许多便利的写法,大大节省了设计者的时间,使得 CSS 的开发变得简单和可维护。所以建议读者学完 CSS 以后,掌握 SASS 语言的用法。



(5) 当你掌握了 SASS 的一些基本入门知识点后，可以趁热打铁，“工欲善其事，必先利其器”，最好再去学习一下 SASS 工具 Compass。Compass 是 SASS 团队成员开发的，是对 SASS 的一个封装，目的是为开发者提供一些丰富的组件以及一些实用的工具模块。由于浏览器对 CSS3 支持的差异性，我们很多时候需要写一堆针对不同浏览器的前缀样式，着实很烦人，而 Compass 帮我们解决了这个问题，我们只需引入相应样式定义即可，Compass 会自动为我们生成针对不同浏览器的样式定义。

(6) 掌握上面提到的全部内容后，即可实现自己想要的效果。但开发页面就像盖大楼一样，每天这样日复一日、年复一年地盖楼，非常烦琐。能不能将大楼里面每一个单独部件模块化，当需要盖楼时就像堆积木一样组合在一起呢？这种思想在 Web 前端开发中同样适用，于是就出现了各种前端框架。在这里，笔者推荐给大家的是 Bootstrap。Bootstrap 是 Twitter 推出的一个开源的用于前端开发的工具包，它是目前最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式页面布局、移动设备优先的 Web 项目，为所有开发者、所有应用场景而设计。Bootstrap 让前端开发更快速、简单，所有开发者都能快速上手，所有设备都可以适配，所有项目都适用。在项目开发过程中，可以借助 Bootstrap 提供的 CSS 样式、组件、JavaScript 插件等快速完成页面布局和样式设置，然后再有针对性地微调样式，这样基于框架进行开发大大缩短了开发周期。所以，能掌握一款好的前端框架是非常有帮助的。

(7) 掌握基础知识后，下一步就是熟练使用这些技术。要想做到这一点，需要多看、多写、多练。通过不断开发项目积累经验，从实战中提升自己解决问题的能力，积累开发素材，摸索创新方法。“君子生非异也，善假于物也”，在开发和学习的过程中还要多浏览一些优秀的网站，善于分析、借鉴其设计思路和布局方法，见多方能识广，进而才可以融会贯通，取他人之长为我所用。同时还要善于使用 Firebug 这样的利器，一方面可以在我们学习过程中帮助我们调试自己的页面，另一方面，我们可以使用 Firebug 方便地查看、分析他人网站的源代码，“偷”也是一种技能。几个项目跟下来，一定可以成为页面制作专家。

注释：截至 2016 年 12 月，Firebug 工具官网通知停止更新和维护，详情请访问：<http://getfirebug.com/>。

1.5.2 第二阶段——编写用户交互功能

通过第一阶段的学习虽然可以完成页面制作，但并不完美，不能算是合格的前端工程师，所以需要继续学习图 1.13 中的第二阶段的内容。现在的 Web 页面都融入了大量的特效，并且多数需要与用户在操作界面上有互动效果。做 HTML5 开发主要使用 JavaScript 语言，JavaScript 是一种在客户端广泛使用的脚本语言，在 JavaScript 当中为我们提供了一些内置函数、对象和 DOM 操作，借助这些内容我们可以来实现一些客户端的特效、验证、交互等，从而使我们的页面看起来不那么呆板。必要时还要学习一些 JavaScript 库，不仅可以简化开

发步骤，还可以处理浏览器之间的兼容问题。jQuery 则是一个免费、开源的轻量级的 JavaScript 库，并且兼容各种浏览器，同时现在有很多基于 jQuery 的插件可供选择，这样当我们实现一些丰富的动态效果时更方便快捷，大大节省了我们的开发时间，提高了开发速度；这也充分体现了 jQuery 的核心宗旨，即“用尽量少的代码完成尽可能多的功能”。如果第二阶段的内容能够完全掌握，再经过几个项目的磨炼，就可以成为真正的前端工程师了。

1.5.3 第三阶段——成为前端工程师

如果第一阶段和第二阶段的内容你已经掌握了（其实也只能算是基础），那么当你做稍大一点的 HTML5 项目开发时，一定会用到第三阶段的技术，如图 1.13 所示。这里笔者将第三阶段的学习内容分为“Web APP 开发”和“Web 游戏开发”两个方向，如果受学习时间限制，读者可以选择其中一个方向学习。

1. Web APP 开发

Web APP 开发成本较低、升级较简单，以及维护比较轻松，它无非就是使用 HTML5、CSS3、JavaScript 等技术做 UI 布局，然后在此基础上再多掌握一些前端框架和封装 APP 的相关机制，服务端技术主要为 PHP、Java 等。目前，Web APP 是非常主流的移动 APP 开发模式，部分项目已经完全取代了传统的 APP 开发形式。

2. Web 游戏开发

电子游戏发展了几十年，玩家越来越被精美的画面、逼真的特效所吸引。所谓 HTML5 游戏，即使用 HTML5 技术所开发的游戏，这种类型的游戏可直接运行于浏览器之中。与传统游戏不同的地方在于，HTML5 游戏具有良好的跨平台与传播性，只需要一个 URL 链接地址，即可进行游戏。不仅如此，还不受时间、地点、设备的限制。目前市场上绝大部分 HTML5 游戏都是基于移动设备的，很少存在基于 PC 设备的 HTML5 游戏。学习 HTML5 游戏制作，关于图像处理方面最成功的新特性莫过于 Canvas 和 WebGL 了，这两者都是用来处理图像和渲染的。所以学习 HTML5 游戏开发，一定要先掌握这两个技术，它们的出现为 HTML5 游戏提供了一种可能——基于 Canvas 和 WebGL，可以依赖浏览器制作出精美的游戏，同时不需要第三方插件的支持。这种技术耦合度是天然的，为 HTML5 游戏提供了坚实的基础。当然，开发 HTML5 游戏也并非全部通过手写基础代码来实现，目前有很多流行的 HTML5 游戏引擎，这也是开发 HTML 游戏前必修的一门技术。笔者推荐国内一款 Egret（白鹭）游戏引擎，其新版本所有功能模块都趋于稳定，通过 Egret 可以开发出复杂、好玩的 HTML5 游戏。



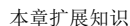
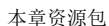
本章小结

学习 HTML5，我们需要了解两个组织：WHATWG（网页超文本应用技术工作小组）和 W3C（万维网联盟）。WHATWG 是一个以推动网络 HTML5 标准为目的而成立的组织，而 W3C 主要为解决 Web 应用中不同平台、技术和开发者带来的不兼容问题，保障 Web 信息的完整流通而制定了一系列标准。HTML5 让 Web 再次回归富客户端，而且更加独立，减少了对第三方插件的依赖；对本地离线存储提供更好的支持；同时增加了新的特殊内容元素，更好地支持 SEO 以及方便视障人士使用；HTML5 还增加了更加智能的表单，给数据校验等控制提供了很好的支持，解决了版本控制、浏览器兼容性和非标准等一系列问题。总的来说，HTML5 的新特性带给开发者的是更友好、更丰富的本地处理 API，更智能、更优雅的 HTML 标签，更强大的本地处理功能，通信也进一步加强。相信由于 HTML5 标准化的支持，未来 Web 技术可以真正在任何端应用，使 Web 应用更加独立。

本章习题

- 关于 HTML5 的说法正确的是（ ）。
 - HTML5 只是对 HTML4.0 的一个简单升级
 - 所有主流浏览器都支持 HTML5
 - HTML5 新增了离线缓存机制
 - HTML5 主要针对移动端进行优化
- 为了标识一个 HTML 文件应该使用的 HTML 标签是（ ）。
 - `<p></p>`
 - `<boby></body>`
 - `<html></html>`
 - `<table></table>`
- 在客户端网页脚本语言中最为通用的是（ ）。
 - JavaScript
 - VB
 - Perl
 - ASP
- 当链接指向下列哪一种文件时，不打开该文件，而是提供给浏览器下载。（ ）
 - ASP
 - HTML
 - ZIP
 - CGI
- 下列不是 HTML5 特有的存储类型的是（ ）。
 - localStorage
 - Cookie
 - Application Cache
 - sessionStorage
- 下列哪项不是 HTML5 的新特性？（ ）
 - 兼容性
 - 合理性
 - 安全性
 - 有插件

11. 简述题: HTML5 存储类型有什么区别?



第2章

HTML5 的基本语法



HTML 也是计算机编程语言，但由于功能简单易用，不涉及业务逻辑，所以算是编程语言中最简单的了。其实学习 HTML 这门语言，就是在学习一个个 HTML 标签的使用，标签的名称和使用不是自定义的，它的功能及用法是已经规定好的，以规范化的方式决定了页面在浏览器中的显示，它们以页面的结构和内容为基础，浏览器会自动对这些标签译码并显示。所以在学习 HTML 标签之前，首先需要了解 HTML 的语言结构和语法构成，本章介绍的语法全部以 HTML5 标准为核心。HTML5 是最新一代的 HTML 标准，和上一代 HTML4.0 相比，只不过是新增一些实用的，废弃一些不常用的和修改一些功能不足的标签及属性。HTML4.0 和 HTML5 其实都是 HTML，HTML 现在是一个持续的动态标准，所以现在学习 HTML 可以直接从 HTML5 开始，不需要先学习 HTML4.0，然后再升级学习 HTML5。当然，在没有完全普及 HTML5 的过渡时代，本章也会对比介绍一下 HTML5 与 HTML4.0 之间的差别。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

2.1 课前准备

在学习一门编程语言之前，需要先了解一下它的用途和功能特性，以及可能用到的一些

专业术语。最重要的是要掌握它的运行原理和开发环境，同时也要选择一款合适的开发工具。学习 HTML 也一样，在开始学习之前，先来了解一下本节内容。

2.1.1 了解Web

Internet 采用超文本和超媒体的信息组织方式，将信息的链接扩展到整个 Internet 上。Web 就是一种超文本信息系统，它的一个主要概念就是超文本链接，使文本不再像一本书一样是固定的、线性的，而是可以从一个位置复制到另一个位置。你可以从中获取更多的信息，也可以跳转到其他主题上。想要了解某一个主题的内容，只需在这个主题上点击一下，即可跳转到包含这一主题的文档上。正是由于这种多链接性，我们才称其为 Web。

1. Web 是图形化的和易于导航的

Web 非常流行的一个很重要的原因，就在于它可以在一个页面上同时显示色彩丰富的图形和文本的性能。在 Web 之前，Internet 上的信息只有文本形式。Web 可以提供将图形、音频、视频信息集合于一体的特性。同时，Web 是非常易于导航的，只需从一个链接跳转到另一个链接，即可在各页面、各站点之间进行浏览。

2. Web 与平台无关

无论是从 Windows 平台、UNIX 平台、Mac OS X 平台，还是其他平台，你都可以通过 Internet 访问 WWW（World Wide Web，万维网），其对你的系统平台没有什么限制。WWW 是基于 Internet 提供的一种界面友好的信息服务，用于检索和阅读链接到 Internet 服务器上的有关内容。该服务利用超文本、超媒体等技术，允许用户通过浏览器检索远程计算机上的文本、图形、声音及视频文件。

3. Web 是分布式的

大量的图形、音频和视频信息会占用相当大的磁盘空间，我们甚至无法预知信息的大小。对于 Web 来说，没有必要把所有信息都放在一起，可以将信息放在不同的站点上，只需要在浏览器中指明这个站点即可，使在物理上并不一定在一个站点的信息在逻辑上一体化。从用户角度来看，这些信息是一体的。

4. Web 是动态的

由于各 Web 站点的信息包含站点本身的信息，信息的提供者可以经常对站点上的信息进行更新，如某个协议的发展状况、公司的广告等。一般各信息站点都尽量保证信息的时间连贯性，所以 Web 站点上的信息是动态的、经常更新的。这一点是由信息的提供者保证的。

5. Web 是交互的

Web 的交互性首先体现在它的超链接上，用户的浏览顺序和所到站点完全由自己决定，



可以从服务器方获得动态的信息。用户通过填写“表单”可以向服务器提交请求，服务器根据用户的请求返回相应信息。

2.1.2 了解HTML

HTML 即超文本标记语言，是一种用来制作超文本文档的简单标记语言。我们在浏览网页时看到的一些丰富的影像、文字、图片等内容，都是通过 HTML 表现出来的。用 HTML 编写的超文本文档称为 HTML 文档，它能独立于各种操作系统平台，一直被用作 WWW（万维网）的信息表示语言。对于网站 Web 开发人员来讲，不涉及 HTML 语言是不可能的。

- 所谓超文本，是因为它不仅是加入文字的文本文件，还可以加入链接、图片、声音、动画、影视等内容的文本文件。使用 HTML 语言描述的文件，需要通过 Web 浏览器显示出效果。HTTP 协议的制定使浏览器在运行超文本时有了统一的规则 and 标准。
- 所谓标记语言，是在纯文本文件里面包含了 HTML 指令代码。这些指令代码并不是一种程序语言，而只是一种排版网页中资料显示位置的标记结构语言，易学、易懂，非常简单。在 HTML 中每个用来作为标签的符号都是一条命令，它告诉浏览器如何显示文本。这些标签均由“<”和“>”符号，以及一个字符串组成。而浏览器的功能是对这些标签进行解析后，显示出文字、图像、动画或播放声音。
- HTML 文件必须使用 .htm 或 .html 作为文件扩展名，推荐使用 .html，是比较安全的做法。

虽然 HTML 语言对于制作一般的网页完全可以胜任，但为了网页的美观和与用户具有交互性的效果，还需要用到 CSS 和 JavaScript 等网页制作技术。CSS 是层叠样式表，是网页页面排版样式标准，能够将格式和结构分离，使浏览器的界面更加友好。JavaScript 是一种描述性脚本语言，和 CSS 一样可以嵌入 HTML 中，在客户端计算机中执行。JavaScript 是具有交互性的 Web 设计的最佳选择，也是浏览器普遍支持的语言。所以，HTML、CSS 和 JavaScript 三项技术结合使用，才是网页设计及制作静态网页的核心。

2.1.3 了解HTML运行原理

HTML 是目前网络上应用最为广泛的语言，也是构成网页文档的主要语言。HTML 只不过是组合成一个文本文件的一系列标签，编写 HTML 代码就是在编辑纯文本文件，但 HTML 是一种规范，需要通过添加特定的标记符来组成描述性文本，可以说明文字、图片、动画、声音、表格、链接等，告诉浏览器如何显示网页，即确定内容的格式。HTML 的结构包括头部、主体两大部分，其中头部描述浏览器所需的信息，而主体则包含所要说明的具体内容。

浏览器按顺序阅读 HTML 文件，然后根据内容周围的 HTML 标记符解析和显示各种内容，这个过程叫作语法分析，如图 2.1 所示。



图 2.1 浏览器解析 HTML 文件显示结果

HTML 是网络的通用语言，是一种简单、通用的全置标记语言，HTML 代码文件在本地和从服务器中下载到本地用浏览器打开效果是一样的。通常开发阶段都是在本地编写的代码自己直接用浏览器打开查看效果，而网站上线则是将所有文件上传到 Web 服务器，所有人都可以通过网址访问 Web 服务器中的网页，将 HTML 文件下载到本地再用浏览器打开浏览网页内容。HTML 中的超文本功能，也就是超链接功能，使网页之间可以链接起来。网页与网页的链接构成了网站，而网站与网站的链接则构成了色彩斑斓的互联网。



2.1.4 如何选择开发工具

有许多可以编辑网页的软件，事实上你不需要用任何专门的软件来建立 HTML 页面，你所需要的只是一个文本编辑器（或文字处理器），如 Office Word、记事本、写字板等。制作页面初学者通常都会选择一个集成开发环境（IDE），如 Dreamweaver，入门快、见效快，在不知不觉中已经完成了页面制作。但是随着学习的深入，你就会发现步入了一种窘境，因为过分依赖 IDE 会导致我们不清楚其实现的本质，知其然但不知其所以然。特别是页面出现 Bug 时，不用工具你便手足无措，更不用提如何进行页面优化及完成一些更高级的应用了。所以，越“聪明”的 IDE，越容易使我们忽略华丽网页背后最本质的代码。这里笔者建议学习阶段最好还是选择简单的工具，如 Vim 或 Notepad++ 等代码编辑工具，只用一些基本的编辑功能，能使开发语言中关键字高亮显示即可。这样就可以学习到开发语言的本质，对学习



非常有帮助,但还是建议在实际开发中多使用功能齐全的 IDE,毕竟能提高一定的开发效率。一些常在学习中使用的网页开发工具如图 2.2 所示。

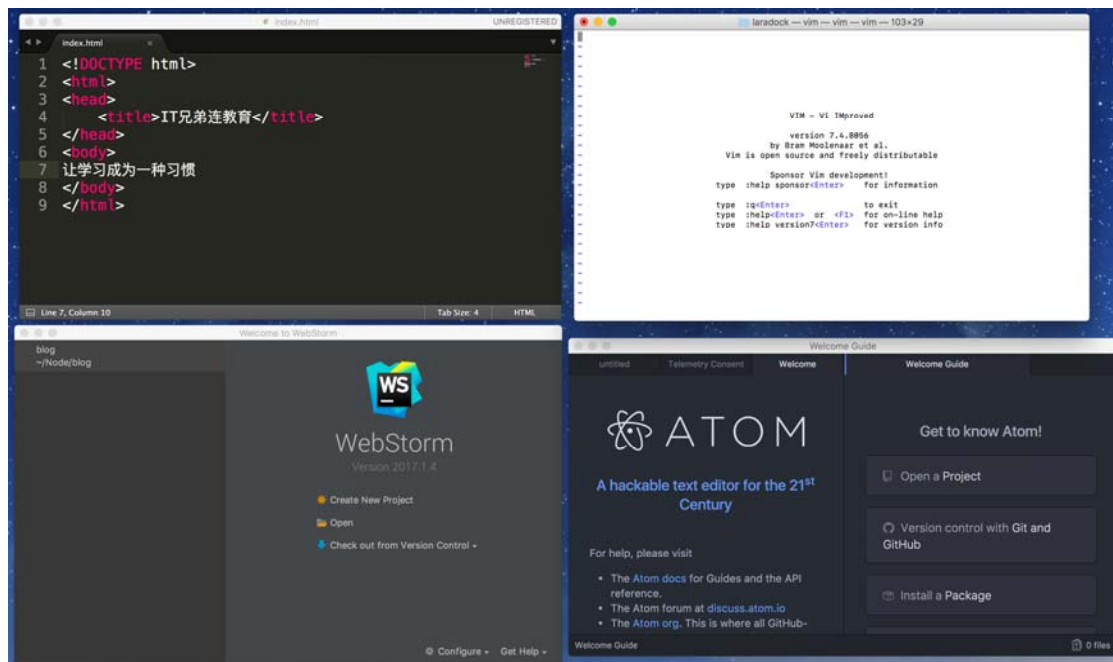


图 2.2 常见的网页开发工具

2.1.5 认识浏览器中的开发者工具

使用浏览器渲染 HTML 代码,和使用音乐播放器听歌曲、使用视频播放器看电影是同样的原理。对于普通用户来说,使用浏览器就是为了浏览网页,但这只用到了浏览器最基本的功能。而浏览器厂商不仅为普通用户考虑,还专门为开发者提供了很多调试页面的工具,对网页开发、升级、维护都非常有帮助。浏览器开发工具一直是 Web 开发者最得力的工具,它能够与 Web 浏览器和谐相处,允许我们在当前窗口中实时操作页面元素、样式和前端程序,以及获取一些其他有用信息。开发工具最大的特点就是很容易使用,但很多开发者往往因为不了解而错过了它们所提供的大部分功能。当然浏览器的种类很多,浏览器内核也不统一,以前开发者使用 Firefox 的一个名叫 Firebug 的扩展来开发和调试网站;但现在,各个浏览器都开发了一套自己的工具,并且每套工具都有自己的优势和劣势。表 2.1 所示为常用浏览器及其开发工具集。

表 2.1 常用浏览器及其开发工具集

Logo	浏览器	开发工具集	存在类型
	Chrome	Developer Tools	集成
	Firefox	Firebug	扩展
	Internet Explorer (IE)	Developer Toolbar	集成
	Opera	Dragonfly	集成
	Safari	Developer Tools	集成（默认关闭）

如果没有这些方便的工具，那么构建一个网站会十分困难。因为本书主要使用谷歌的 Chrome 作为演示浏览器，所以重点讲解 Chrome 中开发者工具的使用。如果读者喜欢使用 Firefox 浏览器，则可以下载安装 Firebug 工具插件。激活开发工具通常按“F12”键（Mac 系统为 Cmd + Option + I），或用鼠标右键单击页面，在弹出的快捷菜单中选择“审查元素”，或者在 Chrome 的工具中找到，如图 2.3 所示。



图 2.3 Chrome 开发者工具的几种启动方式



开发者工具打开后如图 2.4 所示,也可以通过鼠标拖曳进行放大或缩小操作,或通过一些按钮改变位置,以及弹出作为一个独立的窗口存在,当然也可以关闭。



图 2.4 Chrome 开发者工具的样子

通过 Chrome 浏览器中的开发者工具可以查看、编辑页面上的元素,包括 HTML 和 CSS,直接对工具中元素的 HTML 进行编辑,或者删除某个元素,所有的修改都会即时在页面上得到呈现,即修改即时生效。还可以对某个元素进行监听,在 JS 对元素的属性或 HTML 进行修改时,直接触发断点,跳转到对该元素进行修改的 JS 代码处。还可以点击 Network,这里可以看到所有的请求信息,我们点击某一个请求地址,可以看到该请求的详细 HTTP 请求情况,包括 HTTP 请求头、HTTP 响应头、HTTP 返回的内容等信息,对开发、调试都很实用。另外,你可以打开 JavaScript 控制台,做一些 JavaScript 代码的查看或修改操作。除了查看错误信息、打印调试信息、写一些测试脚本,还可以当作 JavaScript API 查看用。

Chrome 不仅简洁、快速,现在 Chrome 的插件也非常丰富。而对于 Web 开发者来说,Chrome 对于 HTML5、CSS3 等一些新标准的支持也是比较完善的,开发者工具非常好用,这就是为什么笔者向 Web 开发者推荐使用 Chrome 的原因。

2.1.6 现在学习HTML5 的方式

目前 HTML 还处于 HTML4.0 与 HTML5 的过渡使用阶段。移动端的 Web 界面开发已经

全面使用 HTML5 技术，而在 PC 端，由于用户升级浏览器周期较长，面临着页面的兼容性问题，以及开发人员对 HTML5 新技术需要一段时间去了解和熟练，所以学习 HTML 必须兼顾这两个版本。可以按版本升级的方式，先学习 HTML4.0 的技术，再延伸学习 HTML5 新增的内容。而 HTML 是一个不断变化的标准，不管是哪个版本，都属于 HTML 技术，所以本书直接学习 HTML5 的标准，当然遇到与 HTML4.0 变化较大的地方也会重点指出。

2.1.7 简单HTML实例制作

学习使用 HTML 制作网页，最好、最快捷的方式就是边学习边做实验，那么我们现在来建立一个简单的范例，非常容易。因为网页文档是纯文本文件，所以只需两个工具（一个是文本编辑器，另一个是浏览器）即可。下面介绍一个简单的 HTML 实例的制作步骤，所有的 HTML 页面的制作都可以参考这个过程。

第一步：打开文本编辑器（推荐 Notepad++，Linux 系统用户可以使用 Vim）。记住，如果你使用的是比较复杂的文字处理器，就应该用“纯文本”或“普通文本”来保存。

第二步：输入页面的主体标记。每个 HTML 页面都要包含这些主体标记，代码如下。

```
1 <!DOCTYPE html>
2 <html lang="en">
3     <head>
4         <meta charset="utf-8">           <!-- 注释：为页面设置字符编码 -->
5         <title>【标题】第一个HTML实例</title>
6     </head>
7
8     <body>
9         <h2>【主题】无序列表制作</h2>
10        <ul contenteditable="true">
11            <li>【列表项】兄弟连IT教育</li>
12            <li>【列表项】兄弟连-猿代码
13            <li>【列表项】兄弟连-兄弟会</li>
14            <li>【列表项】兄弟连-UI-HTML5-PHP学科</li>
15        </ul>
16    </body>
17</html>
```



第三步：将它命名为“xxxx.html”（可以任意命名，扩展名也可以是.htm），如 mypage.html。

第四步：使用浏览器直接打开该文件，则会看到自己做的简单页面，如图 2.5 所示。



图 2.5 第一个 HTML 实例演示

如果希望将该页面发布到 WWW 上，让任何人都可以访问，就需要将其存放到 Web 服务器的文档根目录下。然后在客户端浏览器（Chrome）的地址栏中，通过输入该文件的 URL 进行访问，如 `http://localhost/mypage.html`。页面的显示内容和在本机中直接使用浏览器打开的结果相同，都是通过浏览器解析同样的 HTML 代码后输出的结果。

2.2 HTML 语言的语法

HTML 是文本类型的语言，和其他任何一门语言相比，语法都是最简单的。但在编写 HTML 文件时，必须遵循 HTML 的语法规则。一个完整的 HTML 文件由标题、段落、列表、表格、文本，即嵌入的各种对象所组成，这些逻辑上统一的对象称为元素，HTML 使用标签来描述这些元素。实际上，整个 HTML 文件就是由元素与标签组成的文本文件，可以直接由浏览器解析执行，显示出美妙的网页，而无须编译。当用浏览器打开网页时，浏览器读取网页中的 HTML 代码，分析其语法结构，然后根据解析的结果显示网页内容。正因为如此，网页显示的速度与网页代码的质量有很大关系，保持精简和高效的 HTML 源代码是十分重要的。也可以在浏览器打开的网页中，通过“查看源文件”命令查看网页中的 HTML 代码。

2.2.1 HTML 标签和元素

在 HTML 文件中，是以标签来标记网页结构和显示内容资料的。以“<标签名>”表示标签开始，以“</标签名>”表示结束。大部分标签都是成对出现的，成对的标签也称为容器。在一对标签中，也可以嵌套其他标签。一个 HTML 标签及标签中嵌套的内容就是网页中的一个“HTML 元素”。例如，在<body>和</body>之间的是主体元素；又如，<title>和</title>是标签，而<title>itxdl</title>则是标题元素。也有极少的标签不需要与之配对的结束标签，称为空标签，即空元素，如
、<hr>等。</body>和</title>关闭它们各自的标签。所有的 HTML 标签都要关闭。尽管老版本的 HTML 允许某些标签不关闭，但最新标准要求所有的标签都要关闭。无论如何，闭合标签是一个好习惯。并不是所有的标签都像<html></html>

一样关闭, 有的标签不用绕在内容外面, 它们是自关闭的。比如断行的标签是这样的: `<br />`。需要记住的是, 所有的标签都必须关闭, 以及大部分的内容都在标签之间, 它们的格式是这样的: 起始标签—内容—结束标签。图 2.6 所示为一个 HTML 区块元素。



图 2.6 一个 HTML 区块元素

2.2.2 HTML语法不区分字母大小写

HTML 标签名和属性均不区分字母大小写, 如`<body>`、`<BODY>`或`<Body>`定义的是相同的标记, 但推荐全部使用小写字母书写。在 HTML5 中, 也不区分关键字大小写, 引号也不区分是单引号还是双引号。

2.2.3 HTML标签属性

属性是为 HTML 标签所提供的附加信息, 经常以“名称=值”对的形式出现在 HTML 标签中, 如`<tag name="value">`。大多数 HTML 标签都有自己的一些属性, 要写在起始标签内, 用于进一步改变显示的效果。如果有多个属性, 则使用空格分隔开, 各属性之间无先后次序, 而且 HTML 标签中的每个属性都是可选的, 也都可以省略而采用默认值。属性的值可以用英文的双引号 (") 或单引号 (') 引起来, 也可以不使用引号, 但推荐使用双引号 (W3C 规范)。例如, `<body bgcolor="black" color="#FFFFFF">`标签中使用了两个属性, 分别将`<body>`标签中的内容背景设置为黑色, 文字设置为白色。

2.2.4 HTML颜色值的设置

大多数浏览器都支持颜色名集合, 颜色值是一个关键字或一个 RGB 格式的数字, 在网页中用得很多。仅仅有 16 种颜色名被 W3C 的 HTML 4.0 标准所支持, 分别是 aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white、yellow。如果需要使用其他颜色, 那么需要使用十六进制的颜色值。十六进制的颜色值是由一个十六进制符号来定义的, 这个符号由红色、绿色和蓝色的值组成 (RGB)。每种颜色的最小值是 0 (十六进制: #000), 最大值是 255 (十六进制: #FFF)。也就是说, 每个原色可



有 256 种彩度，故此三原色可混合成 16 777 216 种颜色。应用时需要在每个 RGB 值之前加上 “#” 符号，如 `bgcolor="#00FF00"`。如果使用英文表示颜色值，则可以直接写英文名称，如 `bgcolor="green"`。

2.2.5 HTML文档注释

如果希望在源代码中添加注释，便于阅读，那么可以以 “`<!--`” 开始，以 “`-->`” 结束。HTML 注释的使用如下所示：

```
1 <!-- ----- -->
2 <!--      文件名称: mypage.html      -->
3 <!--      文件说明: 第一个HTML实例  -->
4 <!-- ----- -->
```

注释语句只出现在源代码中，浏览器在解释代码时会忽略注释的内容，而不会在浏览器中显示。这样可以为自己或他人进行注释，或者临时注释掉没有准备好的文档部分，使其不影响后面的代码运行。但需要注意的是，不要在注释中再包含注释，而且注释不能在标签中使用。此外，注释可以包围和隐藏标签。但需要注意的是，在注释掉标签之后，要保证剩余的文本仍然是一个结构完整的 HTML 文档。

提示：除非必要，否则尽量不要在 HTML 代码中使用过多的注释，因为注释的部分需从服务器中下载到本地，会占用带宽从而影响页面加载速度。

2.2.6 HTML代码格式

任何回车或空格在源代码中都不起作用，所以在编写 HTML 代码时，可以使用回车或空格进行代码排版，这样可以使代码更清晰。必须保证严格的缩进规则，以 “Tab” 键为准。

2.2.7 HTML字符实体

一些字符在 HTML 中拥有特殊的含义，例如小于号 (<) 用于定义 HTML 标签的开始，不可以直接在网页中输出。如果我们希望浏览器能正确显示这些有特殊含义的字符，则必须在 HTML 源代码中插入字符实体。

字符实体有三部分：和号 (&)、实体名称或者使用 “#” 符号和一个实体编号、分号 (;)。例如，要在 HTML 文档中显示小于号，我们需要使用 “`<`” 或 “`<`” 实体形式输出。建议使用实体名称而不使用实体编号，好处在于名称相对来说更容易记忆。但需要注意的是，实体名称对大小写敏感。空格是 HTML 中最普通的字符实体，通常情况下，HTML 会裁掉

文档中的空格。例如，在文档中连续输入 10 个空格，那么会被去掉其中的 9 个。如果使用“ ”，则可以在文档中增加空格。还有一些比较常用的 HTML 字符实体，如表 2.2 所示。

表 2.2 常用的 HTML 字符实体

显示结果	描 述	实体名称	实体编号
	空格	 	
<	小于号	<	<
>	大于号	>	>
&	和号	&	&
“	引号	"	"
‘	撇号（IE 不支持）	'	'
¢	分	¢	¢
£	镑	£	£
¥	日元	¥	¥
§	节	§	§
©	版权	©	©
®	注册商标	®	®
×	乘号	×	×
÷	除号	÷	÷

2.3 HTML文档的主体结构

每个页面都是一个独立的 HTML 文档，每个 HTML 文档的主体结构又都是相同的，而且在一个文档中这样的主体结构只能声明一次。可以简单地将 HTML 文档的主体结构分为两部分：一部分是定义文档类型，HTML5 中声明文档类型比以前的版本简单多了，只需要 15 个字符就可以搞定；另一部分则是定义文档主体的结构框架标签，因为标签并不是任意排放的，需要有一定的嵌套规则。就像一棵树按照从树根到树干、树枝、树叶这样的结构生长，而不能在树叶上长出树根来，HTML 文档的结构也是如此。整个文档是一个整体，最外层标签只有一个，第二层标签有两个，这是固定的结构，第三层以后可以任意嵌套，就像一棵倒立的树，如图 2.7 所示。

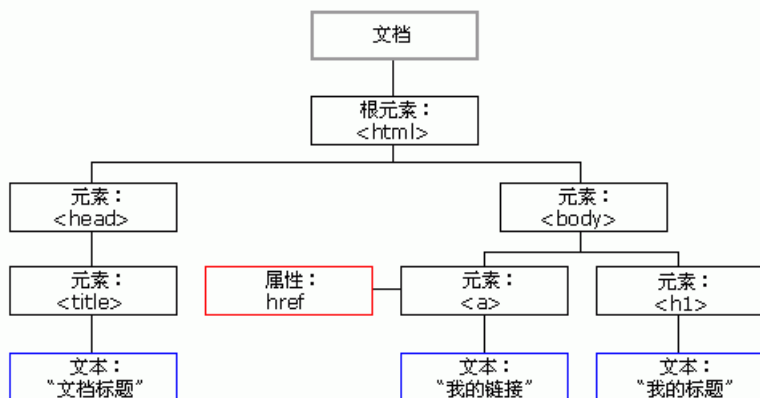


图 2.7 HTML 文档的树形结构

在一个 HTML 文档中，不仅可以通过根节点去寻找到每个子层节点元素，而且从任意一个元素节点出发，都可以通过节点关系找到其他元素。

2.3.1 HTML 文档类型的新定义方式

在编写 HTML5 文档时，要求指定文档类型，以确保浏览器能在 HTML5 的标准模式下进行渲染。在 HTML5 中刻意不使用版本声明，一份文档将会适用于所有版本的 HTML，非常简便。声明方法如下：

```
<!DOCTYPE html>          <!-- 声明没有结束标签，对大小写不敏感 -->
```

声明必须是 HTML 文档的第一行，位于<html>标签之前。另外，声明不是 HTML 标签，它是指示 Web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。在 HTML 4.01 中，<!DOCTYPE>声明引用 DTD，因为 HTML 4.01 基于 SGML，DTD 规定了标记语言的规则，这样浏览器才能正确地呈现内容。HTML5 不基于 SGML，所以不需要引用 DTD。另外，在 HTML5 中只有这一种声明，而在 HTML 4.01 中有三种<!DOCTYPE> 声明：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">          <!-- 第一种：HTML 4.01 Strict -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">             <!-- 第二种 HTML 4.01 Transitional -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">          <!-- 第三种 HTML 4.01 Frameset -->
```

以前的版本中除上面几种声明外，基于 XHTML 不同版本还有好多种，所以在 HTML5 时代，无须再使用上面既麻烦又难记的文档类型，而使用新的 HTML5 文档类型即可，简单明了，这就是 HTML5 的进步。

提示：请始终向 HTML 文档添加<!DOCTYPE>声明，这样浏览器才能获知文档类型。

2.3.2 HTML文档的主体标签

一个 HTML 文档的基本格式需要包含以下几个全局架构元素标签，并将 HTML 代码分为三部分编写，它们可以被看作文档的标准结构，如下所示：

```

1 <!DOCTYPE html>           <!-- 声明新的HTML5文档类型 -->
2 <html>                     <!-- HTML文件开始 -->
3     <head>                 <!-- HTML文件的头部开始 -->
4         <title>文档的标题</title> <!-- HTML文件头部内容标题 -->
5     </head>                <!-- HTML文件的头部结束 -->
6
7     <body>                 <!-- HTML文件主体部分开始 -->
8         文档的内容..... <!-- HTML文件的主体内容部分 -->
9     </body>                <!-- HTML文件的主体结束 -->
10 </html>                   <!-- HTML文件结束 -->

```

本例在网页文件中声明的这几对标签，在每个网页文档中都是唯一的，<head>标签和<body>标签需要嵌套在<html>和</html>标签中。

- 第一部分：<html>和</html>是网页文件的最外层标签，HTML 文档中所有的内容都应该在这两个标签之间。<html>标签告诉浏览器这是 HTML 文档的开始点，</html>标签告诉浏览器这是 HTML 文档的结束点。
- 第二部分：位于<head>和</head>标签之间的文本是头信息，放在<html>标签的最上面使用，头信息不会显示在浏览器窗口中。其主要包括当前页面的一些基本描述语句，用于说明文档的标题和整个文档的一些公共属性，如声明网页的标题和关键字等。每个<head>标签应当包含一个<title>标签以指示文档的标题，它也可以以任意顺序包含<base>、<object>、<link>、<style>、<script>、<meta>标签的任意组合。
- 第三部分：<body>标签是 HTML 文档的主体标签，标签之间的文本是正文内容，用户能够在浏览器主窗口中看到。例如，文字、图片、链接、表单等都需要声明在这个标签中。该标签出现在<head>标签之后。

当然，在 HTML5 新的标准规范中，这些主体标签是可以省略的，浏览器会包容这一点而不会出错，这也是 HTML5 使用灵活的地方。但笔者认为，在编写 HTML5 代码时，没有必要省略它们，保持 HTML 文档结构的完整性，会让可读性更好。

2.4 HTML文档头部标签<head>

HTML 文档的头部标签<head>，主要包括页面的一些基本描述语句，以及 CSS 和 JavaScript，一般都可以定义在头部标签中。它用于包含当前文档的有关信息，如网页标题和



关键字等。通常位于头部的内容都不会在网页上直接显示，而是通过其他方式起作用。在<head>标签中可以使用的标签不多，包含一些常见的，如<title>、<base>、<link>和<meta>等标签。<head>与</head>标签之间必须有<title>标签。

2.4.1 <title>标签

浏览器会以特殊的方式来使用标题，并且通常把它放置在浏览器窗口的标题栏或状态栏中。编写每个页面时，应该给其指定一个标题。HTML 文档的标题使用<title>标签，<title>是<head>标签的子元素，用于将内容显示在浏览器的标题栏中，用以说明文件的用途。当把文档加入用户的链接列表、收藏夹或书签列表时，标题将成为该文档链接的默认名称。每个 HTML 文档都应当有标题，在浏览者保存该网页后成为保存后网页的文件名。另外，搜索引擎在收录该页面时，将网页标题作为搜索的关键词，并将其在搜索引擎页面中作为标题显示。基本语法格式如下：

```
<title> 兄弟连 IT 教育：HTML5 学科</title>          <!-- 在头部定义的标题标签 -->
```

使用实际描述站点内容的标题是非常重要的。例如，站点的主页面不应当只使用“网站首页”标注，而是应当采用能够描述站点内容的语句，通常都是“公司名称+公司产品”。对于优秀的页面标题，访问者在阅读它之后就能够了解该页面的内容，而不需要查看页面的实际内容。

注意：一个文档只能使用一个<title>标签，<title>标签中只能包含关于页面标题的文本，而不能包含其他任何标签。另外，<title> 标签是 <head> 标签中唯一要求包含的东西。

2.4.2 <base>标签

在页面中使用<a>、、<link>、<form> 标签，都需要指定 URL。通常情况下，如果在 URL 中使用相对路径，则浏览器会从当前文档的 URL 中提取相应的标签来填写相对 URL 路径中的空白。而<base>标签为页面上的所有链接规定默认地址或默认目标，浏览器随后将不再使用当前文档的 URL，而使用指定的基本 URL 来解析所有的相对 URL。在网页文档中，所有的相对地址形式的 URL 都是相对于这里定义的基本 URL 而言的。例如，如果在<base>中指定基本 URL 为 http://www.itxd.cn，那么在网页中出现的相对链接“test.html”将对应 http://www.itxd.cn/test.html 的页面。因此，如果网页位置发生变化，可以通过修改<base>来适应这一变化。一篇文档中的<base>标签最多只能有一个，而且必须放于头部，并且应该在任何包含 URL 地址的语句之前。基本语法格式如下：

```
<base href="http://www.itxd.cn/h5/" target="_blank" />      <!-- base 标签使用方法 -->
```

在<base>标签中，href 是必需的属性，规定页面中所有相对链接的基准 URL；而 target 是可选属性，设置在何处打开页面中所有的链接。

2.4.3 <link>标签

<link>标签定义文档与外部资源的关系，最常见的用途是链接样式表。该元素始终是空元素，它仅包含属性，浏览器会分析<link>中的内容，自动读取并加载相应的文件。基本语法格式如下：

```
<link rel="stylesheet" type="text/css" href="style.css" />      <!-- 在头部链接 CSS 文件位置 -->
```

<link>标签只能存在于 head 部分，其出现次数不受限制。关于<link>标签的应用，将在本书后面的 CSS 部分详细介绍。

2.4.4 <meta>标签

<meta>标签可提供有关页面的元信息（meta-information），比如针对搜索引擎和更新频度的描述和关键词，也能够提供文档的作者、描述、编码和语言等多种元信息，但不包含任何内容。该标签位于文档的头部，可以包含任意数量的<meta>标签。该标签的属性通过定义与文档相关联的名称/值对，来定义文件信息的名称、内容等。<meta>是实现元数据的主要标签，通过该标签中的 http-equiv、name、content 属性，可以建立各种各样的效果和功能。基本语法格式如下：

```
<meta name="某个设置值" content="对该设置值进行具体补充说明的信息" />
<meta http-equiv="某个设置值" content="对该设置值进行具体补充说明的信息" />
```

下面给出一段包含<head>标签的源代码，以兄弟连 IT 教育官方网站为例。登录 <http://www.itxdl.cn> 后，通过查看源文件的方式即可找到以下头部信息。

```
4 <head>
5   <!-- meta http-equiv="content-type" content="text/html; charset=GB2312" -->      <!-- meta元素使用 -->
6   <meta charset="UTF-8" />
7   <meta name="keywords" content="PHP培训,PHP视频,HTML5培训,HTML5教程,UI培训" />
8   <meta name="description" content="中国IT培训领跑者，专注PHP培训、PHP教程、
9   HTML5培训、UI培训,培训的人才与企业需求紧密对接，拼教学，论严管，谈素养，
10  比就业，选择IT实力培训机构，兄弟连IT教育，就够了！" />
11  <meta name="generator" content="BroPHP v2.2" />
12  <meta name="author" content="XDL PHP & H5 Team">
13  <meta name="copyright" content="2007-2018 EDU.">
14  <title>兄弟连IT教育-PHP培训_HTML5培训_UI培训</title>      <!-- title元素使用 -->
15
16  <base id="headbase" href="http://www.itxdl.cn/" />      <!-- base元素使用 -->
```



```
17
18 <link rel="stylesheet" type="text/css" href="data/new.css" /> <!-- link元素使用 -->
19 <link rel="icon" href="favicon.ico?v=3" type="image/x-icon" />
20 </head>
```

以上是头部信息的一些基本用法，其中最重要的就是<title>标签及<meta>标签中的 keywords 和 description 属性的设定。因为这两个语句可以让搜索引擎更准确地发现你的站点，吸引更多的人访问。根据现在流行搜索引擎的工作原理，搜索引擎先派机器人自动在万维网上进行搜索，当发现新的网站时，便会检索页面中的 keywords 和 description，并将其加入到自己的数据库，然后再根据关键词的密度对网站进行排序。在 HTML5 中，可以使用对<meta>标签直接追加 charset 属性的方式来指定字符编码，从 HTML5 开始，对于文件的字符编码推荐使用 UTF-8，如下所示：

```
<meta charset="UTF-8"> <!-- HTML5 -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <!-- HTML 以前版本 -->
```

2.5 HTML文档主体标签<body>

在 HTML 的<body>和</body>标签中定义文档的主体，包含文档的所有内容（如文本、超链接、图片、表格和列表等）。<body>标签有自己的属性，设置<body>标签内的属性，可以控制整个页面的显示方式。该标签的基本使用格式如下：

```
7 <body bgcolor="#FFFFE7" color="#FF0000"
8   link="#3300FF" alink="#FF00FF" vlink="#9900FF">
9   <!-- 所有的正文内容都在此声明 -->
10 </body>
```

上面的代码中，<body>标签中的所有属性几乎都是用来设置样式使用的，即“呈现属性”。从 HTML4.01 开始，所有<body>标签的“呈现属性”都可以使用 CSS 统一替代，所以均不被赞成使用。在 HTML5 中当然更不建议使用<body>标签的自身属性设置样式。当然，还有 4 个通用的属性，不仅能够在<body>标签中使用，在任何 HTML 的标签中都可以使用。但这 4 个属性通常只有在结合 CSS 和 JavaScript 时才会使用，如表 2.3 所示。

表 2.3 通用的 HTML 标签的属性

属 性	描 述
id	设定标签的 ID
name	设定标签的名称
class	设定标签样式的类选择器
style	设定标签样式属性

2.6 HTML5 做到了与之前版本的兼容

为了保证 HTML5 能与之前的 HTML 版本最大限度地达到兼容，HTML5 对一些元素标签的省略、boolean 值的属性，以及引号的使用等方面进行了兼顾，确保与之前版本的 HTML 实现兼容。在下面的示例中，将本节介绍的几个 HTML5 新应用方法集成在一起使用：

```
1 <!DOCTYPE html>                                <!-- 在HTML5中声明DTD的方法 -->
2                                                     <!-- 可以全部省略标记html和head及body -->
3 <meta charset="UTF-8">                            <!-- 使用HTML5的方法指定字符编码 -->
4 <title>HTML语法演示</title>
5
6 <br/>这段代码是HTML5语法编写的                <!-- 不允许使用结束标记元素br -->
7 <p>第一段                                       <!-- 可以省略结束标记的元素p -->
8 <p>第二段
9
10 <input type="checkbox" checked>                    <!-- 只写属性不写属性值代表属性为true -->
11 <input type="checkbox">                            <!-- 不写属性代表属性值为false -->
12 <input type="checkbox" checked="checked">          <!-- 属性值=属性名，代表属性为true -->
13 <input type="checkbox" checked="">                 <!-- 属性值=空字符串，代表属性为true -->
14
15 <input type="text">                              <!-- 属性值使用双引号 -->
16 <input type='text'>                             <!-- 属性值使用单引号 -->
17 <input type=text>                               <!-- 属性值不使用引号 -->
```

2.6.1 可以省略标记的元素

元素的标记分为三种情况：不允许写结束标记的元素，可以省略结束标记的元素和可以省略全部标记的元素。不允许写结束标记的元素是指不允许使用开始标记和结束标记将元素括起来的形式。例如，换行标记正确的书写方式为“
”，而“
...</br>”的书写方式就是不标准的。可以省略全部标记的元素，是指该元素可以完全被省略，当然被省略的标记还是以隐式的方式存在的，如“<html>”元素省略不写时还是存在的。针对这三种情况的列举清单如表 2.4 所示。

表 2.4 三种情况列举清单

不允许写结束标记的元素	可以省略结束标记的元素	可以省略全部标记的元素
br、hr、img、input、link、meta、base、param、area、col、command、embed、keygen、source、track、wbr	li、dt、dd、p、option、thead、tbody、tr、td、th、rt、rp、optgroup、colgroup、tfoot	html、head、body、colgroup、tbody

参考上例第 1~8 行代码。



2.6.2 具有boolean值的属性

在 HTML 中，有一些标签的属性，当只写属性名称而不指定属性值时，表示属性值为 true；如果设置该属性值为 false，则不使用该属性即可。另外，要想将属性值设定为 true，也可以将属性名设定为属性值，或将空字符串设定为属性值。例如，<input>标签中的 disabled 与 readonly 就是这样的属性。参考上例第 10~13 行代码。

2.6.3 引号的使用

在 HTML 中使用属性时，属性值可以使用双引号或单引号括起来。在 HTML5 中做了一些改进，当属性值不包括空字符串、“<”、“>”、“=”、单引号、双引号等字符时，属性两边的引号可以省略。参考上例第 15~17 行代码。

2.7

设置IE 9 以下版本浏览器支持HTML5

HTML5 刚发布时，由于各浏览器之间的标准不统一，开发者的时间都浪费在解决 Web 浏览器之间的兼容性上。但由于 W3C 和 WHATWG 对 HTML5 新版本的制定，以及近年来对 HTML5 的使用，再加上各大浏览器鼎力支持，已经有非常丰富的兼容性解决方案，多数 HTML5 应用在老版本的浏览器上也可以正常运行。正是因为保障了兼容性，才能让人们毫不犹豫地用 HTML5 开发网站。HTML5 内部并没有封装一些复杂的、不切实际的功能，而是封装了简单实用的功能。HTML5 内部功能不是革命性的，而是发展性的，并不代表 HTML4.0 创建出来的网站必须全部要重建，只会要求各 Web 浏览器今后能正常运行用 HTML5 开发出来的功能。最新版本的 Safari、Chrome、Firefox 以及 Opera 支持某些 HTML5 特性。IE 9 将支持某些 HTML5 特性，IE 10 将全面支持 HTML5。IE 8 及其以下 IE 版本对 HTML5 标签的支持是有限的，我们可以通过在网页中添加脚本的方式来解决目前 IE 浏览器对 HTML5 部分常用功能支持的问题。让 IE（包括 IE 6）支持 HTML5 元素，我们需要在 HTML 头部添加以下 JavaScript 代码，Opera、Firefox 等其他非 IE 浏览器则会忽视这段代码。

```
<!--[if lt IE9]>
  <script src="http://www.itxdl.cn/h5/html5.js"></script>
<![endif]-->
```

上面这段代码仅会在 IE 浏览器下运行，还有一点需要注意，在页面中调用 html5.js 文件必须添加在页面的<head>标签内，因为 IE 浏览器必须在标签解析前知道这个标签，所以这个.js 文件不能在页面底部调用。另外，页面还需要主体结构完整，像 body 和文档类型都

不能少。如果觉得这个 HTML5 的.js 文件会影响你的网页打开速度,那么可以把 HTML5 的.js 文件直接下载下来,然后上传到自己的服务器单独调用。

随着 HTML5 的发展与普及,越来越多的用户会选择安装支持 HTML5 较好的浏览器使用,到那时开发人员将不用再考虑 IE 9 以下版本的兼容性问题。

本章小结

一个完整的 HTML 文件由标题、段落、列表、表格、文本,即嵌入的各种对象所组成,这些逻辑上统一的对象称为元素。HTML 文档主体结构分为两部分,一部分是定义文档类型,另一部分则是定义文档主体的结构框架标签。一个 HTML 文档需要包含的全局架构元素标签为<html>、<head>、<body>。<html>和</html>是网页文件的最外层标签,<head>和</head>标签之间的文本是头信息,<body>标签是 HTML 文件的主体标签,标签之间的文本是正文内容,是用户能够在浏览器主窗口中看到的。HTML5 对一些元素标签的省略、boolean 值的属性,以及引号的使用等方面进行了兼顾,确保与之前版本的 HTML 实现兼容。

本章习题

1. 下面哪一个不是 HTML5 新增的语义化标签元素? ()
A. section B. head C. article D. aside
2. HTML5 中将 DOCTYPE 分为 () 种。
A. 3 B. 1 C. 2 D. 4
3. 以下标签中不负责组织 HTML 文档基本结构的是 ()。
A. <html> B. <head> C. <body> D. <title>
4. 在 XHTML 文档中, () 是一个必要的元素,它决定了网页文档的显示规则。
A. body B. style C. header D. DOCTYPE
5. HTML5 的正确 DOCTYPE 是 ()。
A. <!DOCTYPE html> B. <!DOCTYPE HTML5>
C. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0//EN" "http://www.w3.org/TR/html5/strict.dtd">
D. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
6. 在 HTML5 中,下列哪个元素不允许写结束标记? ()



- A. br B. li C. rp D. td
7. HTML 的文档注释是以下的哪项? ()
- A. <-- 注释代码 --> B. <!-- 注释代码 --!>
- C. <!-- 注释代码 --> D. <! 注释代码 >
8. 下列哪项可以定义文档关键词, 用于搜索引擎更友好地收录页面? ()
- A. <meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">
- B. <meta name="description" content="Free Web tutorials on HTML and CSS">
- C. <meta name="author" content="Hege Refsnes">
- D. <meta http-equiv="refresh" content="30">
9. 以下说法不正确的是 ()。
- A. HTML5 标准还在制定中
- B. HTML5 兼容以前 HTML4.0 下浏览器
- C. <canvas>标签替代 Flash
- D. 简化的语法
10. HTML5 中不再支持下面哪个标签? ()
- A. <cite> B. <acronym> C. <abbr> D. <base>
11. 简述题: DOCTYPE 有什么作用? 严格模式与混杂模式如何区分? 它们有何意义?



本章习题及其答案



本章资源包



本章扩展知识

第3章

HTML5 文字版面和编辑标签



网页中的信息主要以文本为主，可以通过字体、大小、颜色、底纹、边框等来设置文本的属性。文字版面的编辑包括文本标签和格式标签两种，在浏览器中显示的文字内容和格式都要在<body>标签中编写。文字是网页设计最基础的部分，一个标准的文字页面可以起到传达信息的作用。对文字的格式化，通常可以使用两种方式：一种方式是直接使用 HTML 标签，另一种方式是使用 CSS。利用 CSS 可以对文本的格式进行精确控制，使用 HTML 标签则更有利于搜索引擎抓取。本章主要介绍一些和页面排版相关的标签。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

3.1 HTML基础标签

HTML 标签有很多，为了方便学习和使用，我们可以将这些标签按类别细分一下。基础标签则是页面制作经常使用的一些标签，包括上一章介绍的结构标签，都属于基础标签。本节介绍的基础标签如表 3.1 所示。



表 3.1 HTML 基础标签

标 签	描 述
<h1>~<h6>	定义 HTML 标题
 	换行标签，完成文字的紧凑显示。如果有多个换行，则可以连续使用多个 标签
<p>	换段落标签，即换行之后插入一个空行，然后在下一行显示其后的文本。可以使用成对的<p>标签来包含段落，也可以使用单独的<p>标签来划分段落
<hr>	水平分隔线标签，用于段落与段落之间的分隔

3.1.1 标题标签<h1>~<h6>

HTML 定义<h1></h1>到<h6></h6> 6 个标签对，分别用于设计不同大小字体的标题，字体由大到小，<h1>最大，<h6>则最小。需要注意的是，<h7>及其他数字标签则不是 HTML 标签。这 6 个标题标签属于块级元素，浏览器会自动在标题的前后添加空行。页面中的标题对整个页面来说很重要，应该将这些标签只用于标题，而不要仅仅是为了产生粗体或大号的文本而使用标题。相反，我们应当使用层叠样式表定义来达到漂亮的显示效果。这些标题标签对 SEO（搜索引擎优化）是很友好的，搜索引擎使用标题为你网页的结构和内容编制进行索引，而且用户也会通过标题来快速浏览你的网页，所以用标题来呈现文档结构是很重要的。应该将<h1>用作主标题（最重要的），其次是<h2>（次重要的），然后是<h3>，以此类推。需要注意的是，<h1>在一个页面中只能使用一次，如果使用多次，搜索引擎就会认为你在“作弊”。标题标签的使用和演示结果如图 3.1 所示。

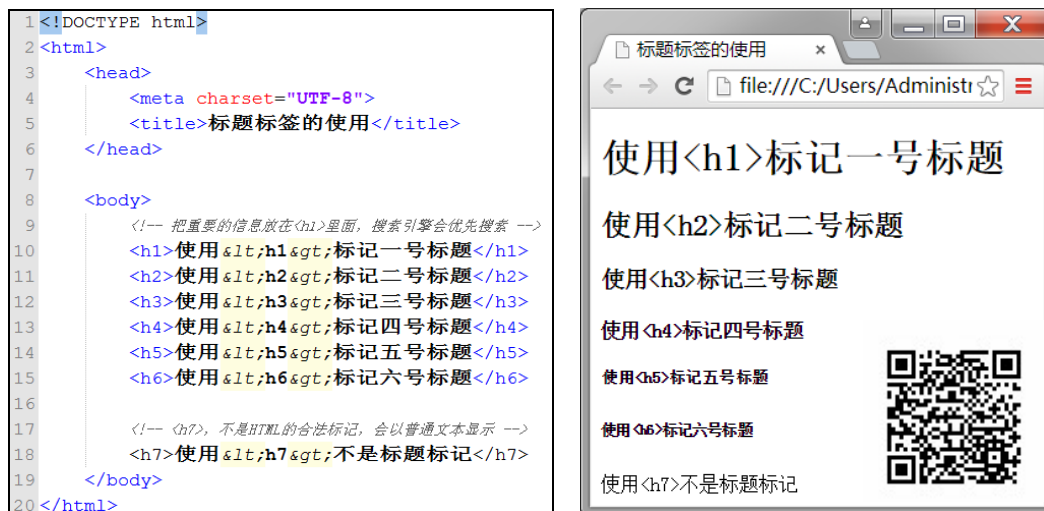


图 3.1 标题标签的使用和演示结果

标题标签也有四个可选的属性，即 left、center、right 和 justify，用来规定标题中文本的排列，但不推荐使用，可以使用样式替代它们。

3.1.2 换行标签
和段落标签<p>

可插入一个简单的换行符，
 标签是空标签（意味着它没有结束标签，因此这是错误的：
</br>）。关于 HTML 中的换行标签
和段落标签<p></p>的区别是，换行标签
只是回车，<p></p>则是分段。因为<p>标签属于块级元素，它的前后会有比较大的空白，而
标签属于行级元素，它只是简单的换行，前后没有空白。有一个与
相对的标签<nobr></nobr>，它表示不换行，通常用于防止浏览器将程序代码等不需要换行的地方自动换行。换行标签和段落标签的演示结果如图 3.2 所示。

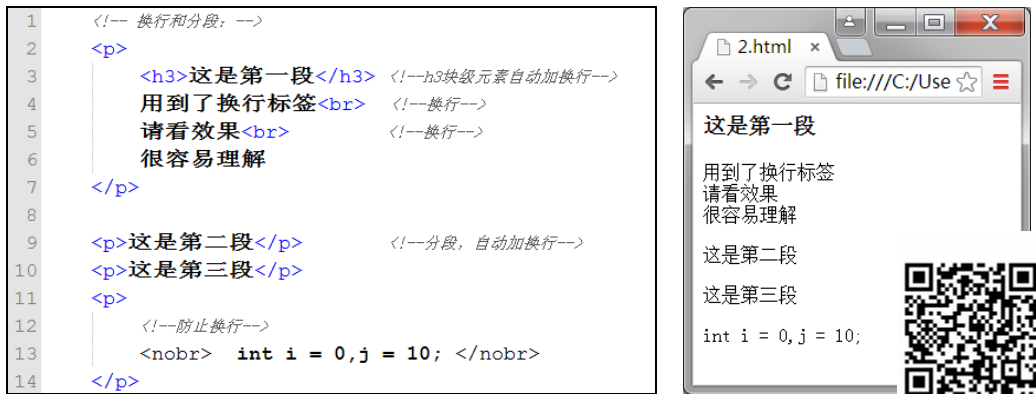


图 3.2 换行标签和段落标签的演示结果

3.1.3 水平分隔线标签<hr>

<hr>标签在 HTML 页面中创建水平线，可以在视觉上将文档分隔成各个部分，常常用来分隔文章中的小节。<hr>标签和
标签一样是单标签，没有结束标签。<hr>标签可以设置宽度（width）和高度（height），这两个属性又分别有像素和百分比的表示方式。此外，<hr>标签还有 size（厚度）、color（颜色）和 noshade（无阴影）属性。<hr>标签的所有呈现属性均不被赞成使用，都可以使用样式替代它们。

3.1.4 格式标签

格式标签用于定义网页中文本的布局，如缩进、位置、换行控制、列表等。当浏览器遇到这些标签时，就会按标签的格式显示网页。例如，
标签具有换行的功能，当浏览器遇



到
标签时，就会把标签后的文本从新的一行开始显示。常见的格式标签如表 3.2 所示。

表 3.2 常见的 HTML 格式标签

标 签	描 述
 	换行标签，完成文字的紧凑显示。如果有多个换行，则可以连续使用多个 标签
<p>	换段落标签，即换行之后插入一个空行，然后在下一行显示其后的文本。可以使用成对的<p>标签来包含段落，也可以使用单独的<p>标签来划分段落
<center>	居中对齐标签，使段落或文字相对于上一层标签居中对齐
<pre>	预格式化标签，保留文字在源代码中的格式，页面中显示的和源代码中书写的格式效果完全一致
	列表项目的标签，每个列表项使用一个 标签
	无序列表，使用 作为无序的声明，使用 作为每个项目的起始。没有顺序号，使用缩进的形式表示一定的层次性
	有序列表，可以显示特定的项目顺序，与无序列表 的使用方式基本相同
<hr>	水平分隔线标签，用于段落与段落之间的分隔

在本例中，简单应用表 3.2 中介绍的格式标签，这些标签全部需要声明在<body>标签中使用，代码如下：

```
1 <html>
2   <head><title>格式标记测试网页</title></head>
3   <body>
4       <p>一段文本</p>
5       <center>这行文本在网页中居中显示</center>
6       <pre>
7           上边
8           左边      右边
9           下边
10      </pre>
11      <hr>
12      无顺序列表：
13      <ul>
14          <li>第一项</li>
15          <li>第二项</li>
16          <li>第三项</li>
17      </ul>
18      <hr>
19      有顺序列表：
20      <ol>
21          <li>第一项</li>
22          <li>第二项</li>
23          <li>第三项</li>
24      </ol>
25  </body>
26 </html>
```

在头标记中输出网页的标题
主体标记的开始
使用段落标记输出一行文本
居中标记设置一段文本居中显示
预定义标记保留源代码格式输出
保留这些文字在源代码中的格式
保留这些文字在源代码中的格式
保留这些文字在源代码中的格式
保留这些文字在源代码中的格式
在段落与段落之间输出的分隔
在页面中输出一行文本
无顺序列表的开始标记
无顺序列表中第一个列表项目
无顺序列表中第二个列表项目
无顺序列表中第三个列表项目
无顺序列表的结束标记
在段落与段落之间输出的分隔
在页面中输出一行文本
有顺序列表的开始标记
有顺序列表中第一个列表项目
有顺序列表中第二个列表项目
有顺序列表中第三个列表项目
有顺序列表的结束标记
主体标记的结束标记



上面的示例代码在浏览器中的输出结果如图 3.3 所示。



图 3.3 HTML 格式标签输出结果演示

3.1.5 文本标签

在网页中，为了强调某一部分文字，或者为了让文字有变化，HTML 提供了一些标签产生这些效果。常见的文本标签如表 3.3 所示。

表 3.3 常见的 HTML 文本标签

标 签	描 述
<h1>	标题字标签，n 为 1~6，定义了 6 级标题，而且会自动换行插入一个空行
	粗体字标签
<i>	斜体字标签
<u>	下画线字体标签
<sub>	文字下标字体标签
<sup>	文字上标字体标签
	字体标签，可以通过该标签指定字体大小、颜色、字体等信息
<tt>	打字机文字
<cite>	用于引证、举例，通常为斜体字
	表示强调，通常为斜体字
	表示强调，通常为粗体字
<small>	小型字体标签
<big>	大型字体标签



在本例中，简单应用表 3.3 中介绍的文本标签，与前面介绍的格式标签使用方式类似，也需要全部声明在<body>标签中使用，代码如下：

```
1 <html>
2   <head>
3     <title>文本标记测试网页</title>
4   </head>
5
6   <body>
7     <h1>使用<h1>输出一号标题字体</h1>
8     <h2>使用<h2>输出二号标题字体</h2>
9     <h3>使用<h3>输出三号标题字体</h3>
10    <h4>使用<h4>输出四号标题字体</h4>
11    <h5>使用<h5>输出五号标题字体</h5>
12    <h6>使用<h6>输出六号标题字体</h6>
13
14    <font face="楷体_GB2312" color="red" size="5">绝对字号大小为5的红色楷体字</font><br>
15    <font face="宋体" color="green" size="+3">相对字号大小为+3的绿色宋体字</font><br>
16    <font face="黑体" color="blue" size="-1">相对字号大小为-1的蓝色黑体字</font><br>
17
18    <b>使用<b>标记输出粗体字</b><br>
19    <u>使用<u>标记输出带有下划线的文字</u><br>
20    <i>使用<i>标记输出斜体字</i><br>
21
22    <tt>使用<tt>标记输出打印机文字</tt><br>
23    <cite>使用<cite>标记输出引证、举例的斜体字</cite><br>
24    <em>使用<em>标记输出强调的斜体字</em><br>
25    <strong>使用<strong>标记输出强调的粗体字</strong><br>
26    <small>使用<small>标记输出小型的字体</small><br>
27    <big>使用<big>标记输出大型的字体</big><br>
28  </body>
29</html>
```



上面的示例代码在浏览器中的显示结果如图 3.4 所示。

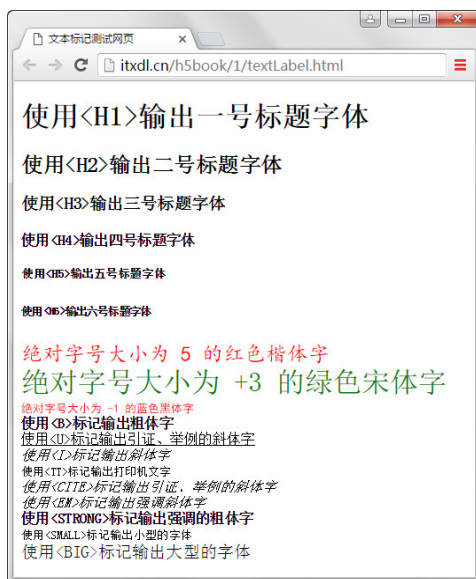


图 3.4 HTML 文本标签输出结果演示

3.2 使用HTML表格

表格在网站中应用得非常广泛，使用它可以方便、灵活地排版，很多动态大型网站也都是借助表格排版的，但现在都使用 DIV+CSS 进行页面布局。表格可以把相互关联的信息元素集中定位，使浏览页面的人一目了然。在 HTML 文档中，表格需要通过表格标签<table>、行标签<tr>、单元格标签<th>或<td>等标签按一定的关系嵌套共同完成，<tr>、<td>标签必须定义在<table>标签之间。表格的基本结构如下：

```
1 <table>                                <!-- 用于定义一个表格开始标记          -->
2   <caption> 表格名称 </caption>         <!-- 用于定义表格的标题标记          -->
3
4   <tr>                                  <!-- 定义一行标记，可以在内部建立多组单元格 -->
5       <th> 表头单元格 </th>            <!-- 定义表头单元格，文字将以粗体居中显示 -->
6   </tr>                                  <!-- 用于定义行标记结束标记          -->
7
8   <tr>                                  <!-- 定义一行标记，可以在内部建立多组单元格 -->
9       <td> 单元格 </td>                <!-- 定义一个单元格标记          -->
10  </tr>                                  <!-- 用于定义行标记结束          -->
11 </table>                                <!-- 用于定义一个表格结束标记          -->
```

1. <table>标签

该标签用于定义一张表格。在一张基本的表格中，必须包含一组<table>标签、一组<tr>标签和一组<td>或<th>标签。在<table>标签中，可以使用表 3.4 中给出的一些常用的可选属性。

表 3.4 HTML 中<table>标签的常用属性

属 性	描 述
align	该属性用于指定表格在上一层标签中的显示位置。该属性的值为 left、center 或 right，分别表示表格居左、居中或居右的对齐方式，默认值为 left
border	该属性用于指定表格外边线的宽度，单位为像素
width	表示表格的宽度，它可以是像素值，也可以是上层标签的百分比表示
height	表示表格的高度，它可以是像素值，也可以是上层标签的百分比表示
cellspacing	该属性设定表格单元格之间的间距，单位为像素，默认值为 2
cellpadding	该属性设定表格单元格内容与单元格边框之间的距离，单位为像素

2. <caption>标签

该标签用于定义表格标题的位置，可以由 align 和 valign 属性来设置。align 属性设置标题位于文档的左、中或右；valign 属性设置标题位于表格的上方或下方。默认属性是标题位于表格的上方中间位置。该标签应放在<table>标签内，在表格行标签<tr>之前。



3. <tr>标签

该标签定义一行标签。表格是由行和列组成的，一张表格由几行组成就要有几个行标签<tr>，一组行标签内可以建立多组由<td>或<th>标签所定义的单元格。行标签用它的属性值来修饰，属性都是可选的。例如，align 属性设置行内容的水平对齐；valign 属性设置行内容的垂直对齐；bgcolor 属性设置行的背景颜色。

4. <th>和<td>标签

<th>和<td>都是插入单元格的标签，这两个标签必须嵌套在<tr>标签内，是成对出现的。<th>用于表头标签，表头标签一般位于首行或首列，标签之间的内容就是位于该单元格内的标题内容，其中的文字以粗体居中显示。数据标签<td>就是该单元格中的具体数据内容。<th>和<td>标签的属性相同，可以使用表 3.5 中给出的一些常用的可选属性。

表 3.5 表格单元格标签的常用属性

属 性	描 述
width/height	单元格的宽和高，接受绝对值（如 80）及相对值（如 80%）
align	单元格内字画等的摆放贴，位置（水平），可选值为：left、center、right
valign	单元格内字画等的摆放贴，位置（垂直），可选值为：top、middle、bottom
rowspan	该属性设定当前单元格所跨越的行数
colspan	该属性设定当前单元格所跨越的列数

5. 综合实例

在本例中，使用表格在网页中输出一个学员信息列表，包括学员基本信息和学员成绩信息，代码如下：

```
1 <html>
2   <head><title>TABLE 表格的使用示例</title></head>
3   <body>
4     <table width="80%" border="1" cellpadding="3" cellspacing="0" align="center">
5       <caption><h1>学员表</h1></caption>           <!-- 设置表格标题位置 -->
6
7       <tr><th colspan="3"> 学员基本信息</th><th colspan="2">成 绩</th></tr>
8       <tr><th>姓 名</th><th>性 别</th><th>专 业</th><th>课 程</th><th>分 数</th></tr>
9
10      <tr align="center">
11        <td>小 峰</td>
12        <td>男</td>
13        <td rowspan="2">计算机</td>
14        <td rowspan="2">PHP 开发</td>
15        <td>86</td>
16      </tr>
17      <tr align="center">
18        <td>小 影</td>
19        <td>女</td>
```



```
20      <td>98</td>
21    </tr>
22  </table>
23 </body>
24 </html>
```

```
<!-- 设置第二行第五个单元格 -->
<!-- 第二行结束标记 -->
<!-- 表格结束标记 -->
<!-- 主体标记的结束标记 -->
<!-- HTML文档的结束标记 -->
```

本例在浏览器中的显示效果如图 3.5 所示。



图 3.5 HTML 表格标签演示结果

3.3 HTML 框架结构

使用 HTML 框架结构可以把一个浏览器窗口划分为若干个小窗口，每个窗口可以显示不同的 URL 网页，每个框架中的网页相互独立。这样不仅可以非常方便地在浏览器中同时浏览不同的页面效果，而且还可以非常方便地完成导航工作。如果所有的框架标签要放在一个 HTML 文档中，则这个 HTML 页面的文档主体标签<body>被框架集标签<frameset>取代，然后通过<frameset>的子窗口标签<frame>定义每个子窗口和子窗口的页面属性。子窗口的排列遵循从左到右、从上到下的次序规则。

1. 划分框架

使用<frameset>标签决定如何划分框架，该标签中有 cols 属性和 rows 属性。使用 cols 属性表示按列分布框架，使用 rows 属性表示按行分布框架。必须使用<frame>标签设定每个小窗口中的网页，该标签中有 src 属性为每个 URL 值指定一个 HTML 文件（这个文件必须事先做好）地址，地址路径可以使用绝对路径或相对路径，这个文件将载入相应的窗口中。如果希望在同一个浏览器窗口中，将其既按照行来分布框架，又按照列来分布框架，则可以将<frameset>标签嵌套使用形成嵌套框架。<frameset>标签常用的属性如表 3.6 所示。



表 3.6 HTML 的<frameset>标签常用的属性

属 性	描 述
cols	用“像素数”和“%”分隔左右窗口，“*”表示剩余部分
rows	用“像素数”和“%”分隔上下窗口，“*”表示剩余部分
frameborder	指定是否显示边框：“0”代表不显示边框，“1”代表显示边框
border	设置边框粗细，默认值是 5 像素

2. 子窗口<frame>标签的设定

<frame>是一个单标签，该标签必须放在框架集标签<frameset>中使用。<frameset>设置了几个子窗口，就必须对应几个<frame>标签，而且每个<frame>标签内还必须使用 src 属性设定一个网页文件。其常用属性如表 3.7 所示。

表 3.7 HTML 的<frame>标签常用的属性

属 性	描 述
src	指示加载的 URL 文件的地址
name	指示框架名称，是链接标记的 target 所要的参数
noresize	指示不能调整窗口的大小，省略此项时则可调整
scrolling	指示是否要滚动条，auto 表示根据需要自动出现，yes 表示有，no 表示无
frameborder	指示是否要边框，1 表示显示边框，0 表示不显示（不提倡用 yes 或 no）
border	设置边框粗细

3. 窗口的名称和链接

如果要在窗口中做链接，就必须对每个子窗口命名，以便被用于窗口间的链接。在窗口的链接中使用 target 属性，就可以将被链接的内容放置到想要放置的窗口内。在下面的例子中，通过框架技术并使用窗口的名称和链接实现后台首页模型，如图 3.6 所示。



图 3.6 HTML 的框架集标签演示结果

在图 3.6 提供的网站后台管理平台界面模型中，当单击“大类管理”选项时，改变左边菜单中的页面。当单击左边的菜单链接时，改变右边主体页面的内容。在文件 index.html 中划分框架的代码如下：

主窗体文件 index.html

```

1 <html>
2   <head><title>使用框架定义后台管理平台模型</title></head>
3   <frameset rows="80,*" frameborder="1" border="5">
4     <frame src="top.html" name="top" scrolling="no" noresize/>
5     <frameset cols="200,*">
6       <frame src="menu.html" name="menu" scrolling="no" noresize/>
7       <frame src="main.html" name="main" noresize />
8     </frameset>
9   </frameset>
10 </html>

```

<!-- 网页开始标记 -->
 <!-- 设置网页标题 -->
 <!-- 划分上下两行 -->
 <!-- 顶部大类窗体 -->
 <!-- 划分左右两列 -->
 <!-- 左边菜单窗体 -->
 <!-- 右边内容窗体 -->
 <!-- 内层框架结束 -->
 <!-- 外层框架结束 -->

在上面的示例代码中，先将窗体分为上下两行，并将顶部窗体命名为 top，设置 80 像素的高度。然后将下部的窗体分为左右两个窗体，分别设置为 200 像素和使用“*”表示剩余部分，并分别命名为 menu 和 main。网页文件 index.html 的文档主体标签<body>被框架集标签<frameset>取代，所以不能在这个框架文件中再有<body>的内容。然后通过<frameset>的子窗口标签<frame>定义每个子窗口，并通过子窗口的属性 src 分别加载 top.html、menu.html 和 main.html 3 个页面文件。这 3 个页面文件的源代码如下：



顶部设置大类选项窗体文件 top.html

```
1 <html>
2   <head><title>无标题文档</title></head>
3
4   <body>
5       <center><h2>后台管理平台</h2></center>
6       <p>
7           <a href="menu.html" target="menu">大类管理（一）</a> ||
8           <a href="menu2.html" target="menu">大类管理（二）</a> ||
9           <a href="menu3.html" target="menu">大类管理（三）</a>
10      </p>
11  </body>
12</html>
```

左边设置菜单选项窗体文件 menu.html

```
1 <html>
2   <head><title>无标题文档</title></head>
3   <body>
4       <center><h3>大类管理（一） 菜单</h3></center>
5       <p>
6           <a href="main.html" target="main">管理选项1</a><br>
7           <a href="main2.html" target="main">管理选项2</a><br>
8           <a href="main3.html" target="main">管理选项3</a><br>
9       </p>
10  </body>
11</html>
```

右边设置内容窗体文件 main.html

```
1 <html>
2   <head><title>无标题文档</title></head>
3   <body>
4       <center><h4>大类管理（一） > 管理选项1 > 内容设置</h4></center>
5   </body>
6</html>
```

在 top.html 文件的每个链接中,通过 target 属性设置左边菜单窗体名称 menu,当单击“大类管理”选项时,链接文件就会在左边窗体中显示。同样,在 menu.html 文件的每个链接中,通过 target 属性设置右边窗体的名称 main,当操作每个菜单选项时,对应的链接文件就会在名为 main 的窗体中加载。当然,在本例中还需要为每个大类管理选项定义一个独立的菜单页面,也需要为每个菜单项定义唯一的内容页面。

本章小结

HTML 基础标签是在页面制作中经常使用的一些标签,包括标题标签<h1>~<h6>、换行标签
、段落标签<p>、水平分隔线标签<hr>、格式标签及文本标签。表格可以把相互关联的信息元素集中定位,使浏览页面的人一目了然。表格需要通过表格标签<table>、行标签<tr>、单元格标签<th>或<td>等标签按一定的关系嵌套共同完成。通过使用框架,可以在

同一个浏览器窗口中显示不止一个页面。每个 HTML 文档称为一个框架，并且每个框架都独立于其他框架。

本章习题

1. 以下哪个是水平线标签？（ ）
A. `<br>` B. `<hr>` C. `<pre>` D. `<frame>`
2. 以下是 HTML5 新增的标签的是（ ）。
A. `<aside>` B. `<isindex>` C. `<samp>` D. `<s>`
3. 下列的 HTML 标签中，哪个是一级标题？（ ）
A. `<h6>` B. `<head>` C. `<heading>` D. `<h1>`
4. 以下选项中，哪一项全部是表格标签？（ ）
A. `<table>` `<head>` `<tfoot>` B. `<table>` `<tr>` `<td>`
C. `<table>` `<tr>` `<tt>` D. `<thead>` `<body>` `<tr>`
5. 在 HTML5 中，哪个标签用于组合标题元素？（ ）
A. `<group>` B. `<header>` C. `<headings>` D. `<hgroup>`
6. 下列哪一项属性用来设置单元格垂直的位置？（ ）
A. `align` B. `valign` C. `colspan` D. `rowspan`
7. 在一个框架的属性面板中，不能设置下面哪一项？（ ）
A. 源文件 B. 边框颜色 C. 边框宽度 D. 滚动条
8. 下面哪一个属性不是文本的标签属性？（ ）
A. `nbsp` B. `align` C. `color` D. `face`
9. 下面哪一项是换行标签？（ ）
A. `<body>` B. `<font>` C. `<br>` D. `<p>`
10. 以下标签中，用于定义一个单元格的是（ ）。
A. `<td> </td>` B. `<tr>...</tr>`
C. `<table>...</table>` D. `<caption>...</caption>`
11. 简述题：`<frameset>`标签的优缺点有哪些？



本章习题及其答案



本章资源包



本章扩展知识

第4章

多媒体应用



元素——音频（audio）和视频（video）。

多媒体来自多种不同的格式。它可以是你听到或看到的任何内容，如文字、图片、音乐、音效、录音、电影、动画等。在互联网上，经常会发现嵌入网页中的多媒体元素，现代浏览器已支持多种多媒体格式。在本章中，可以了解不同的多媒体格式，以及如何在网页中使用它们。本章分为三部分，包括创建图像和链接，如何制作 HTML 图像地图，以及 HTML5 新增的多媒体播放



本章二维码

本章二维码里面包括：

- （1）本章的学习视频；
- （2）本章所有实例演示结果；
- （3）本章习题及其答案；
- （4）本章资源包（包括本章所有代码）下载；
- （5）本章的扩展知识。

4.1 创建图像和链接

图像和超链接极大地丰富了 HTML 文档的表现形式。我们看到的丰富多彩的 Web 页面，都是因为有了图像。而超链接是网站的灵魂，可以实现访问其他 Web 页面的功能。在互联网上，图像和链接则是通过 URL 唯一确定信息资源的位置。URL 分为绝对 URL 和相对 URL。当所需资源在远程主机上时，需要使用绝对 URL；而当资源在本机上时，可以只提供文件名部分，这就是相对 URL。

4.1.1 URL概述

我们在浏览器的地址栏中输入的网站地址叫作 URL (Uniform Resource Locator, 统一资源定位符), 是对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示, 是互联网上标准资源的地址。互联网上的每个文件都有一个唯一的 URL, 它包含的信息指出文件的位置及浏览器应该怎么处理它。URL 的格式为:

```
http://<IP 地址>[/端口号]/[路径][?<查询信息>]
```

就像每家每户都有一个门牌地址一样, 每个网页也都有一个 Internet 地址, 即 URL。当用户在浏览器的地址栏中输入一个 URL 或是单击一个超链接时, URL 就确定了要浏览的地址, 然后通过 HTTP 将 Web 服务器上站点的网页代码提取出来, 并翻译成漂亮的网页。例如, `http://www.ydma.cn/book/index.html`, 它的含义如下。

- `http://`: 代表超文本传输协议, 通知 ydma.cn 服务器显示 Web 页面, 通常不用输入。
- `www`: 代表一台 Web (万维网) 服务器。
- `ydma.cn/`: 这是装有网页的服务器的域名, 或站点服务器的名称。
- `book/`: 是该服务器上的子目录, 就好像我们的文件夹。
- `index.html`: 是文件夹中的一个 HTML 文件, 也就是网页。
- 如果使用默认端口 80 可以不写, 如果使用非 80 端口则必须在 URL 中指定。

URL 的第一部分“`http://`”表示的是要访问的文件的类型。有时也使用 `ftp`, 意为文件传输协议, 主要用来传输软件和大文件 (许多做软件下载的网站都使用 `ftp` 作为下载的网址); 还有用 `telenet` (远程登录) 的, 主要用于远程交谈, 以及文件调用等, 意思是浏览器正在阅读本地盘外的一个文件, 而不是一台远程计算机。

4.1.2 插入图片

在 Web 页面中, 通常使用 3 种格式的图片, 即 JPEG、GIF 和 PNG。通过使用 `<img>` 标签在浏览器中显示一张图像, 其语法格式如下:

```
    <!-- 在网页中插入图片 -->
```

上面代码给出了 `<img>` 标签常用的一些属性, 其中 `src` 属性用于指定一个包含 URL 路径名的图像文件, `alt` 属性用于定义一个字符串。`<img>` 的 `alt` 属性不可缺少, 它有 3 个作用: 当浏览该网页时, 如果图像下载完成, 将鼠标放在该图像上, 稍等片刻, 鼠标旁边则会出现这个属性指定的提示文字; 如果图像没有被下载, 在图像的位置上就会显示该属性指定的提示文字; 搜索引擎可以通过这个属性的文字抓取该图片。`width` 和 `height` 两个属性分别用于



指定图像的宽度和高度，单位为像素；如果需要等比例缩放，则只需使用一个即可。border 属性用于指定图像边界的宽度，单位为像素。 标签在网页中的使用如下所示：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>图片插入实例</title>           <!-- 在标题栏输出网页标题 -->
5     <meta charset="utf-8">
6 </head>
7 <body>                                       <!-- 网页主体部分开始 -->
8     <!-- 使用alt属性的图片 -->
9     
10    <!-- 设置图片的边框宽度为2px -->
11    
12    <!-- 只设置宽度，高度自动适应 -->
13    
14    <!-- 图片高度和宽度一起设置 -->
15    
16 </body>                                     <!-- 网页主体部分结束 -->
17 </html>
```



本例在网页中使用标签分别插入 4 张图片，并且都使用 src 属性通过相对 URL 指定图像文件的位置。图 4.1 是插入图片后页面的显示效果，默认情况下文字和图片处于同一行。



图 4.1 使用 HTML 图像标签插入图片的显示结果

4.1.3 建立锚点和超链接

Web 上的网页是互相链接的，单击被称为超链接的文本或图形，就可以链接到其他页面。超文本具有链接能力，可层层链接相关文件，这种具有超链接能力的操作，称为超链接。超

链接除了可链接文本，还可以链接各种媒体，如声音、图像、动画，通过它们，我们可以享受丰富多彩的多媒体世界。链接文档中的特定位置也称为锚点。在浏览页面时，如果页面很长，则需要不断地拖动滚动条，给浏览带来不便。如果浏览者想要从头阅读到尾，又可以选择自己感兴趣的部分阅读，那么这种效果可以通过两个锚点间的一种链接关系来实现。定义锚点和超链接都使用[<a>](#)标签，其基本语法格式如下：

```
<a href="URL" name="name" target="target">链接文字</a>      <!-- 在网页中定义锚点和超链接 -->
```

上面代码给出了[<a>](#)标签常用的一些属性，其中 href 属性指定所链接文件的 URL 路径，name 属性指定页面的锚点名称。如果需要链接到对应的锚点位置，则需要在锚点名前加一个“#”字符。target 属性指定要打开的链接所使用的浏览器窗口名称。可以使用自定义的窗口名称，也可以使用下面 4 个内置的窗口名称。

- [_self](#)：在当前窗口中打开链接文件，是默认值。
- [_blank](#)：开启一个新的窗口打开链接文件。
- [_parent](#)：在父级窗口中打开文件，常用于框架页面。
- [_top](#)：在顶层窗口中打开文件，常用于框架页面。

下面是一段定义锚点和超链接的示例代码，使用了文字链接和图片链接，以及通过相对 URL 和绝对 URL 分别链接本地文件和远程文件。

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>建立锚点和超链接示例</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <a name="anchor_one" />      <!-- 在本页声明一个锚点，名为anchor_one -->
9
10  <a href="http://www.itxd1.cn/index.php">
11    以绝对URL指定链接文件位置
12  </a><br>
13  <a href="link.html" target="_blank">
14    在新窗口中打开以相对URL指定的文件位置
15  </a><br>
16  <a href="link.html">
17    
18  </a><br>
19
20  <a href="#anchor_one">链接到当前页面的锚点位置</a><br>
21  <a href="link.html#anchor_one">链接到其他页面中的锚点位置</a><br>
22 </body>
23 </html>
```



将包含上述代码的 HTML 文件在浏览器中打开，效果如图 4.2 所示。



图 4.2 锚点和超链接

4.2 HTML 图像地图

图像地图是带有可点击区域的图像，通常情况下，每个区域是一个相关的超级链接。点击某个区域，就会到达相关的链接，也可以通过图像地图实现图片切换效果。

4.2.1 什么是图像地图

把一幅图像分成多个区域，每个区域指向不同的 URL 地址。例如，将一幅中国地图按照省市划分为若干个区域，这些区域就被称为热点。点击热点区域，就可以链接到与相应的省市有关的页面，这就是图像地图。

4.2.2 图像地图如何制作

(1) 首先必须定义出图像上的各个热点区域的形状、位置坐标及指向的 URL 地址等信息，这个过程叫作图像热点映射。图像热点映射需要使用 `<map name=mapname></map>` 标签对进行说明，其中 `name` 属性为该图像热点映射指定了一个名称。

(2) 图像热点映射中的各个区域用 `<area>` 标签说明，`<area>` 标签的格式为：`<area shape="形状" coords="坐标" href="URL">`。href 部分也可以用 `nohref` 替换，表示在该区域单击鼠标无效。`<area>` 标签还可以有一个 `target` 属性，用来指明浏览器在哪个窗口或者帧中显示 href 属性所指向的网页资源。`<area>` 标签的属性及描述如表 4.1 所示。

(3) 定义好了图像热点之后，接着就要在 `<img>` 图像标签中增加一个名为 `usemap` 的属

性设置。usemap 属性指定该图像被用作图像地图，其设置值为所使用的图像热点映射名称，格式为：在<map>标签中的 name 属性设置值前多加一个“#”字符，如。

表 4.1 <area> 标签的属性及描述

属 性	描 述
alt	定义此区域的替换文本
coords	坐标值，定义可点击区域（对鼠标敏感区域）的坐标
href	URL，定义此区域的目标 URL
shape	定义区域的形状。可选值有 default（全部区域）、rect（矩形）、circle（圆形）、poly（多边形）
target	规定在何处打开 href 属性指定的目标 URL。可选值有_blank、_parent、_self、_top

4.2.3 实现图像地图

我们为下面的“itxdl.png”图片划分一个地图，利用标签的 usemap 属性设定图片热点，为图片划分 4 个区域（左上、右上、左下、右下），然后给热点设定相应的链接。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>图像地图</title>
5   <meta charset="utf-8">
6   <style>
7     img{ display:block; margin:20px auto; width: 516px; height: 245px }
8   </style>
9 </head>
10 <body>
11   <!-- 为itxdl.png映射为4个区域，
12        当用户单击某一个区域时，将链接到不同的文档中 -->
13   <!-- <img>标签的usemap属性值为#map，#后面的map是自定义图像地图的名称 -->
14   
15   <map name="map">
16     <!-- 分别被地图分区设定为矩形、坐标和链接地址 -->
17     <area shape="rect" coords="0,0,258,122" target="_blank"
18           href="http://www.itxdl.cn/topleft.html" alt="topleft" />
19     <area shape="rect" coords="122,0,516,122" alt="topright"
20           href="http://www.itxdl.cn/topright.html" target="_blank" />
21     <area shape="rect" coords="0,122,258,245" target="_blank"
22           href="http://www.itxdl.cn/bottomleft.html" alt="bottomleft" />
23     <area shape="rect" coords="258,122,516,245" target="_blank"
24           href="http://www.itxdl.cn/bottomright.html" alt="bottomright" />
25   </map>
26 </body>
27 </html>
```





将包含上述代码的 HTML 文件在浏览器中打开，鼠标移动到图像地图的左上角区域，会在浏览器的左下角位置显示该区域的 URL 地址：www.itxdl.cn/topleft.html。点击该区域跳转到该链接，效果如图 4.3 所示。



图 4.3 图像地图展示

上述代码表明，我们为“itxdl.png”链接了一个被平分为 4 个矩形区域的图像热点。图 4.3 只是第一个区块的展示，同理，第二个区块在图像的右上角，第三个区块在图像的左下角，第四个区块在图像的右下角。除了将图像热点设为矩形，我们也可以将它们设为圆形或多边形，但是每种形状的坐标设定不相同。shape 属性的设置说明如表 4.2 所示。

表 4.2 shape 属性的设置说明

属 性	描 述
rect	定义一个矩形区域，coords 属性设置值为左上角、右下角的坐标，各个坐标之间用逗号隔开
poly	定义一个多边形区域，coords 属性设置值为多边形各个顶点的坐标值
circle	定义一个圆形区域，coords 属性设置值为圆心坐标及半径，前两个参数分别为圆心的横、纵坐标，第三个参数为半径

4.3 新增多媒体播放元素

在 HTML5 之前，要在网站上展示视频、音频、动画等多媒体信息，除了使用第三方自主开发的播放器，使用最多的工具应该算是 Flash 了，但是它们都需要在浏览器中安装各种插件才能使用，有时速度很慢。HTML5 的出现使这一局面得到了改观。在 HTML5 中，提供了音频的标准接口，多媒体播放再也不需要安装插件了，只需要一个支持 HTML5 的浏览

器即可。本节介绍 audio 和 video 这两个 HTML5 新增加的元素，它们分别用来处理音频与视频数据。目前 Safari 3 以上、Firefox 4 以上、Opera 10 以上，以及 Google Chrome 3.0 以上的浏览器都实现了对这两个媒体元素的支持。以 audio 元素为例，只要把播放音频的 URL 地址指定给该元素的 src 属性即可。代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>音频audio</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <audio src="itxdl.mp3" controls>您的浏览器不支持音频audio功能</audio>
9   <audio controls>
10     <source src="itxdl.mp3" type="audio/mp3" >
11     <source src="itxdl.ogg" type="audio/ogg" >
12     <source src="itxdl.wav" type="audio/wav" >
13     您的浏览器不支持音频audio功能
14   </audio>
15 </body>
16 </html>

```



将包含上述代码的 HTML 文件在浏览器中打开，如果浏览器支持音频功能，则出现音频播放器，否则页面显示“您的浏览器不支持音频 audio 功能”。效果如图 4.4 所示。

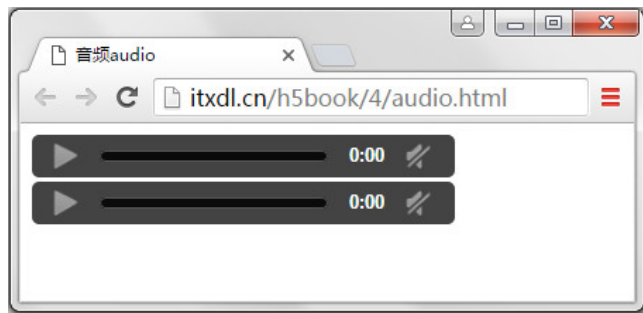


图 4.4 音频<audio>标签

视频 video 元素的使用方法和 audio 相似，只要设定好元素的长、宽等属性即可。代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>视频video</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <!-- 视频标签 -->
9   <video src="itxdl.mp4" controls loop width="400" height="400">
10     您的浏览器不支持视频video功能

```





```
11     </video>
12     <video controls>
13         <source src="itxdl.mp4" type="video/mp4">
14         <source src="test.webm" type="video/webm" >
15         <source src="test.ogv" type="video/ogg" >
16         您的浏览器不支持视频video功能
17     </video>
18 </body>
19 </html>
```

将包含上述代码的 HTML 文件在浏览器中打开，如果浏览器支持视频功能，则出现视频播放器，否则页面显示“您的浏览器不支持视频 video 功能”。效果如图 4.5 所示。



图 4.5 视频<video>标签

各浏览器对于各种媒体类型及编码格式的支持情况各不相同，可以通过 source 元素来为同一个媒体数据指定多种播放格式与编码方式，以确保浏览器可以选择一种自己支持的播放格式进行播放。浏览器的选择顺序为代码中的书写顺序，它会从上向下判断自己对该播放格式是否支持，选到为止。各浏览器对于编码格式的支持情况如表 4.3 所示。

表 4.3 各浏览器对于编码格式的支持情况

	IE 9	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis		√	√	√	
MP3	√			√	√
WAV		√	√		√

audio 和 video 元素通过一些常用属性的使用，可以定义和扩展一些功能，它们所具有的属性大致相同。两个元素常用的属性如表 4.4 所示。

表 4.4 audio 和 video 元素常见属性说明

属性名	值	说 明
src	url	在该属性中指定媒体数据的 URL 地址
autoplay	autoplay	在该属性中指定媒体是否在页面加载后自动播放
preload	none metadata auto（默认）	在该属性中指定视频或音频数据是否预加载。如果使用预加载，则浏览器会预先将数据进行缓冲，这样可以加快播放的速度。有三个可选择值： ➤ none 表示不进行预加载。 ➤ metadata 表示只预加载媒体的元数据（媒体字节数、第一帧、播放列表、持续时间等）。 ➤ auto 表示预加载全部视频或音频
poster	图片 url	video 元素独有属性，当视频不可用时，向用户展示一幅替代的图片
loop	loop	如果出现该属性，则当媒体文件完成播放后再次开始播放
controls	controls	指定是否为视频或音频添加浏览器自带的播放用的控制条，控制条中包含播放、暂停等按钮
width/height	pixels	video 元素独有属性，设置视频的宽度与高度，以像素为单位

除了表 4.4 中提供的属性，还有一些属性、方法和事件可用，但需要结合 JavaScript 来实现一些特定的效果。例如，通过 error 属性可以处理出现的错误；使用 networkState 属性读取当前网络状态；使用 play 方法来播放媒体；使用 load 方法重新载入媒体进行播放；通过 timeupdate 事件来通知当前播放位置的改变；结合 JavaScript 来显示当前的播放进度等。

本章小结

在互联网上，图像和链接则是通过 URL 唯一确定信息资源的位置。URL 分为绝对 URL 和相对 URL。通过使用标签在浏览器中显示一幅图像。超文本具有链接能力，可层层链接相关文件，这种具有超链接能力的操作，称为超链接。链接文档中的特定位置也称为锚点，定义锚点和超链接都使用<a>标签。图像地图是带有可点击区域的图像，每个区域是一个相关的超级链接。audio 和 video 这两个 HTML5 新增加的元素，它们分别用来处理音频与视频数据，使得多媒体播放再也不需要安装插件了。

本章习题

- 1. audio 元素中 src 属性的作用是（ ）。



- A. 提供播放、暂停和音量控件
- B. 循环播放
- C. 制定要播放音频的 URL
- D. 插入一段替换内容

2. 以下哪个不是 HTML5 中 a 元素的属性? ()

- A. type
- B. download
- C. name
- D. rel

3. 下列关于<a>标签的说法错误的是 ()。

- A. <a>标签是超链接, 可以使用 href 属性使其指向另一个资源
- B. 当点击<a>标签时触发的提交为 GET 提交
- C. <a>标签可以嵌套标签, 使图片变为一个可点击的超链接
- D. <a>标签可以指向一张图片, 从而在该位置显示一张图片

4. HTML5 不支持的视频格式是 ()。

- A. ogg
- B. MP4
- C. FLV
- D. WebM

5. 以下关于 video 的说法正确的是 ()。

A. 当前 video 元素支持三种视频格式, 其中 WebM = 带有 Theora 视频编码和 Vorbis 音频编码的 WebM 文件

B. source 元素可以添加多个, 具体播放哪个由浏览器决定

C. video 内使用 img 展示有视频封面

D. loop 属性可以使媒介文件循环播放

6. 用于播放 HTML5 视频文件的正确 HTML5 元素是 ()。

- A. <movie>
- B. <media>
- C. <video>
- D. <audio>

7. 在下列的 HTML 中, 哪个可以产生超链接? ()

- A. 兄弟连 IT 教育
- B. 兄弟连 IT 教育
- C. <a>https://www.itxdl.cn
- D. 兄弟连 IT 教育

8. 如何在新窗口打开链接? ()

- A.
- B.
- C.
- D.

9. 在下列的 HTML 中, 哪个可以插入图像? ()

- A.
- B.
- C. <image src=" itxdl.gif">
- D. itxdl.gif

10. 以下说法正确的是 ()。

A. <a>标签是页面链接标签, 只能用来链接到其他页面

- B. <a>标签的 href 属性用于指定要链接的地址
 - C. <a>标签是页面链接标签，只能用来链接到本页面的其他位置
 - D. <a>标签的 src 属性用于指定要链接的地址
11. 简述题：除了 audio 和 video，HTML5 还有哪些媒体标签？



本章习题及其答案



本章资源包



本章扩展知识

第5章

HTML5 表单



表单在网页应用中十分重要，基本上任何一个网站都必须使用到表单元素，所以表单的美观和易于交互对于网站设计来说就变得十分重要。HTML5 对目前 Web 表单进行了全面提升，使得我们使用表单更加智能。它在保持了简单易用的特性的同时，还增加了许多内置的控件或者控件属性来满足用户的需求，提升用户体验，并且同时减少了开发人员的编程。HTML5 主要对表单的以下方面进行了改进：更自由的表单结构、多样的输入类型、新增的表单属性。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

5.1 HTML表单中的get和post方法

超文本传输协议（HTTP）的设计目的是保证客户机与服务器之间的通信。HTTP 的工作方式是客户机与服务器之间的请求—应答协议。在客户机和服务器之间进行请求—应答时，两种经常被用到的方法是：get 和 post。

当用户在 HTML 表单（HTML Form）中输入信息并提交之后，有两种方法将信息传送到 Web 服务器（Web Server）。一种方法是通过 URL，另一种方法是在 HTTP Request 的 body

中。针对前一种方法，我们使用 HTML Form 的 `method="get"`，而针对后一种方法我们使用 HTML Form 中的 `method="post"`。

5.1.1 get方法

`get` 方法的作用是从指定的资源请求数据。当 HTML 表单中的 `method` 属性的值为“`get`”时，被提交后，URL 发生了改变，在 URL 中显示 HTML Form 参数的 `name/value` 值。通过下面的案例来看看 `get` 是如何提交 form 数据的，代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>HTML表单中的get方法</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <!-- HTML表单通过get方法提交，URL地址改变 -->
9   <form action="get.php" method="get">
10     Name: <input type="text" name="name" />
11     age:  <input type="text" name="age" />
12     <input type="submit" value="提交" />
13   </form>
14 </body>
15 </html>
```

将包含上述代码的 HTML 文件在浏览器中打开，并为表单填充数据后提交，我们可以看到地址栏中的 URL 地址发生了改变，效果如图 5.1 所示。

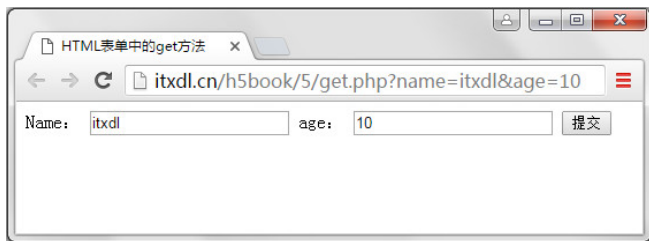


图 5.1 HTML 表单的 `get` 方法

通过图 5.1，我们可以看到浏览器地址栏中的 URL 是：`http://www.itxdl.cn/h5book/5/get.php?name=itxdl&age=10`。注意 `get.php` 后面的拼接字符串 `?name=itxdl&age=10`，这里有两对 `name/value` 数据，前面加一个问号，每对 `name/value` 数据用“`&`”符号隔开。

注：这里的 `get` 方法和 `post` 方法都需要提交到服务器上进行数据的接收。Form 表单的 `action` 属性设置提交到服务器的地址，如 `action="http://www.itxdl.cn/h5book/5/get.php"`。

由于本案例本身存放在服务器上的地址是：域名下的 `h5book/5`，所以案例中就省略了路径“`http://www.itxdl.cn/h5book/5`”。



5.1.2 post方法

post 方法的作用是向指定的资源提交要被处理的数据。跟 get 方法不一样的是,当 HTML 表单中的 method 属性的值为“post”时,被提交后,URL 不会改变。也就是说,不在 URL 中显示 HTML Form 的数据。

5.1.3 HTML表单中get和post的区别

一般在浏览器中输入网址访问资源都是通过 get 方法,在 form 提交中,可以通过 method 指定提交方式为 get 或者 post,默认为 get 提交。让我们来看看这两种提交方式的区别。

1. 请求的数据

(1) get 请求:请求的数据会附在 URL 之后(把数据放置在 HTTP 协议头中),以“?”分隔 URL 和传输数据,多个参数用“&”符号连接。

(2) post 请求:把提交的数据放置在 HTTP 包的包体中。

因此, get 提交的数据会在地址栏中显示出来,而 post 提交时,地址栏不改变。

2. 传输数据的大小

(1) get 请求:特定的浏览器和服务器对 URL 长度有限制,如 IE 对 URL 长度的限制是 2083 字节。对于其他浏览器,如 Netscape、Firefox 等,理论上没有限制,其限制取决于操作系统的支持。

(2) post 请求:由于不是通过 URL 传值,理论上数据不受限。但实际各个 Web 服务器会规定对 post 提交的数据大小进行限制,Apache、IIS7 都有各自的配置。

HTTP 协议没有对传输的数据大小进行限制,HTTP 规范也没有对 URL 长度进行限制。

3. 安全性

(1) get 请求:可被缓存,会保留在浏览器历史记录中,可被收藏为书签。

(2) post 请求:不会被缓存,也不会保留在浏览器历史记录中,不可被收藏为书签。

post 的安全性比 get 的安全性高。通过 get 提交数据,若表单数据包含用户名和密码,那么用户名和密码将明文出现在 URL 上,而且登录页面也可能被浏览器缓存,当其他人查看浏览器的历史记录时,就可以轻松拿到你的账号和密码了。除此之外,使用 get 提交数据还可能造成 CSRF (Cross-site Request Forgery) 攻击。所以, get 请求不应该在处理敏感数据时使用。

5.2 HTML 表单设计

表单是 PHP 程序中最常使用的收集站点访问者信息的数据输入界面。通过表单浏览器获取用户的输入数据，并传送到 Web 服务器的脚本程序中，以各种方式进行处理。在表单中提供了多种输入方式，包括文本输入域、单选或多选按钮、下拉式列表域等。表单是网页上由<form>标签定义的一个特定区域，而表单的各种输入域可以由<input>、<select>和<textarea> 3 个标签定义。

5.2.1 表单标签<form>

一个表单用<form></form>标签来创建，即定义表单的开始和结束位置，在开始和结束标签之间的一切定义都属于表单的内容。单击“提交”按钮时，提交的也是表单范围之内的内容。另外，在<form>标签中需要携带表单的相关信息，如处理表单的脚本程序的位置、提交表单的方法等。这些信息对于浏览者是不可见的，但对于处理表单却起着决定性的作用。该标签的常用属性如表 5.1 所示。

表 5.1 HTML 表单标签<form>的常用属性

属 性	描 述
name	表单名称
method	该属性用来定义处理程序从表单中获得信息的方式，可取值为 get 和 post 中的一个。get 方法是将表单内容附加在 URL 地址后面，所以对提交信息的长度进行了限制，不可以超过 8192 个字符，同时 get 方法不具有保密性，不适合处理如信用卡卡号等要求保密的内容，而且不能传送非 ASCII 的字符。post 方法是将用户在表单中填写的数据包含在表单的主体中，一起传送到服务器上的处理程序中，不会在浏览器的地址栏中显示提交的信息。这种方式传送的数据没有限制。默认为 get 方法
action	该属性的值是处理程序的程序名（绝对或相对 URL），如果这个属性是空值，则当前文档的 URL 将被使用。当用户提交表单时，服务器将执行 URL 中的程序（如 PHP 程序）
enctype	设置表单资料的编码方式
target	该属性和链接中的同名属性类似，用来指定目标窗口或目标帧

<form> 标签中必须加 action 属性，并且不能为空。例如，<form action="login.php" method="post">。如果不需要使用 action 属性，那么也必须定义：<form action="no">。



5.2.2 文本域和密码域

在<form>标签内定义的<input>标签具有重要的地位，该标签是单个标签，没有结束标记。<input type="">标签用来定义一个用户输入区，用户可以在其中输入信息。<input>标签中共提供了 9 种类型的输入区域，具体是哪一种类型由 type 属性来决定。文本和密码输入域是一个单行文本框，它们基本相似，唯一不同的是，用户在密码域中输入的字符并不以原样显示，而是将每个字符用“*”代替。文本和密码输入域的基本语法格式如下：

```
<input type="text" name="field_name" value="field_value" size="n" maxlength="n">      <!-- 输入域 -->
<input type="password" name="field_name" value="field_value" size="n" maxlength="n">  <!-- 密码域 -->
```

这些属性的含义如表 5.2 所示。

表 5.2 HTML 中<input>标签的常用属性

属 性	描 述
type	该属性的值为“text”，表示这是一个文本输入域。如果该属性的值为“password”，则表示这是一个密码输入域
name	定义控件名称
value	指定控件初始值，该值就是浏览器被打开时在文本框中的内容
size	指定控件宽度，表示该文本输入框所能显示的最大字符数
maxlength	表示该文本输入框允许用户输入的最大字符数

5.2.3 提交、重置和普通按钮

在<input>标签中，当 type 属性值为“submit”时，表示这是一个提交按钮，单击“提交”按钮后，可以实现表单内容的提交；当 type 属性为“reset”时，表示这是一个重置按钮，单击“重置”按钮后，表单的内容将恢复为默认值；当 type 属性为“button”时，表示这是一个普通按钮，并不实现任何功能，需要和 JavaScript 等脚本语言一起使用。button 按钮必须定义在<form>标签之间，否则 Netscape 浏览器不支持。这 3 种按钮的基本语法格式如下：

```
<input type="submit" name="field_name" value="field_value">      <!-- 提交按钮 -->
<input type="reset" name="field_name" value="field_value">      <!-- 重置按钮 -->
<input type="button" name="field_name" value="field_value">      <!-- 普通按钮 -->
```

5.2.4 单选按钮和复选框

单选按钮和复选框都有“选中”和“未选中”两种状态。同一组单选按钮如果有多个选择框，则选择框之间是相互排斥的，只允许用户选择其中一个。复选框和单选按钮的区别是，复选框允许用户同时选中同一表单中的多个或全部选项，当然，也可以只选其中的一个选项。单选按钮和复选框只有在“选中”时，数据才能被提交到服务器端。其语法格式如下：

```
<input type="radio" name="field_name" value="field_value" checked>      <!-- 单选按钮 -->
<input type="checkbox" name="field_name" value="field_value" checked>      <!-- 复选框 -->
```

在<input>标签中，当 type 属性值为“checkbox”时，表示这是一个复选框输入域；当 type 属性值为“radio”时，则表示这是一个单选按钮输入域。但在同一组中的多个单选按钮名称必须相同，它们之间才能相互排斥。单选按钮和复选框都可以通过 checked 属性设置为“选中”状态。

5.2.5 隐藏域

隐藏域不会在表单中显示。如果需要在页面之间传递重要数据，则在<input>标签中设置 type 属性值为“hidden”，建立一个隐藏域。name 和 value 属性是必需的，用来表示隐藏域的名字和值。基本的语法格式如下：

```
<input type="hidden" name="field_name" value="field_value">      <!-- 隐藏表单域 -->
```

5.2.6 多行文本域

多行文本域提供一个可以输入多行文本的区域，在该区域中没有输入字符长度的限制。在<form>标签中使用<textarea>标签制作多行文本域。基本的语法格式如下：

```
<textarea name="name" rows="value" cols="value" value="value">      <!-- 多行文本域开始标记 -->
... ..                                                                <!-- 在多行文本域设置默认值 -->
</textarea>                                                          <!-- 多行文本域结束标记 -->
```

在该标签中使用 name 属性指定多行文本域的名称；通过 rows 和 cols 两个属性分别指定多行文本域中显示字符的行数和列数，单位是字符个数。

5.2.7 菜单下拉列表域

在<form>标签中使用<select>标签创建一个菜单下拉列表域，该标签具有 multiple、size 和 name 属性。其中 multiple 属性不用赋值，直接加入标签中即可使用，加入了此属性后列



表域就成为可多选的列表。size 属性用来设置列表的高度，默认值为 1，若没有设置 multiple 属性，显示的将是一个下拉式的列表域。而 name 属性定义此列表框的名称，与前面介绍的 name 属性作用相同。基本的语法格式如下：

```
<select name="name" size="value" multiple>          <!-- 菜单下拉列表域开始标记 -->
  <option value="value" selected>选项</option>        <!-- 指定列表域中第一个选项 -->
  <option value="value">选项</option>                <!-- 指定列表域中第二个选项 -->
  ... ..                                             <!-- 可以指定更多的列表选项 -->
</select>                                           <!-- 菜单下拉列表域结束标记 -->
```

<option>标签用来指定列表域中的一个选项，需要放在<select></select>标签对之间。此标签具有 selected 和 value 属性，selected 属性用来指定默认的选项，value 属性用来给<option>标签指定的那一个选项赋值。这个值是要传送到服务器上的，服务器正是通过调用<select>区域的名字的 value 属性来获得该区域选中的数据项的。

5.2.8 综合实例

在本例中，笔者将使用前面介绍的大部分表单内容，制作 LAMP 兄弟连学员基本信息输出界面。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>兄弟连学员信息</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <table align="center" width="500" border="0" cellpadding="0">
9     <caption align="center"><h2>兄弟连学员基本信息</h2></caption>
10    <form action="server.php" method="post">
11      <tr>                                <!-- 使用输入域定义姓名输入框 -->
12        <th>姓名：</th>
13        <td><input type="text" name="username" size="20"></td>
14      </tr>
15      <tr>                                <!-- 使用单选按钮域定义性别输入框 -->
16        <th>性别：</th>
17        <td>
18          <input type="radio" name="sex" value="1" checked />男
19          <input type="radio" name="sex" value="2" />女
20          <input type="radio" name="sex" value="0" />保密
21        </td>
22      </tr>
23      <tr>                                <!-- 使用下拉列表域定义学历输入框 -->
24        <th>学历：</th>
25        <td>
26          <select name="edu">
27            <option>--请选择--</option>
28            <option value="1">高中</option>
29            <option value="2">大专</option>
30            <option value="3">本科</option>
```

```

31         <option value="4">研究生</option>
32         <option value="5">其他</option>
33     </select>
34 </td>
35 </tr>
36 <tr>
37     <th>选修课程: </th>
38     <td>
39         <input type="checkbox" name="course[]" value="4">Linux
40         <input type="checkbox" name="course[]" value="5">Apache
41         <input type="checkbox" name="course[]" value="6">MySQL
42         <input type="checkbox" name="course[]" value="7">PHP
43     </td>
44 </tr>
45 <tr>
46     <th>自我评价: </th>
47     <td>
48         <textarea name="eval" rows="4" cols="40"></textarea>
49     </td>
50 </tr>
51 <tr>
52     <td colspan="2" align="center">
53         <input type="submit" name="submit" value="提交" />
54         <input type="reset" name="reset" value="重置" />
55     </td>
56 </tr>
57 </form>
58 </table>
59 </body>
60 </html>

```



本例在浏览器中的显示效果如图 5.2 所示。

图 5.2 HTML 表单标签的演示结果



5.3 HTML5 新增表单元素

HTML5 有一些新增的表单元素：`<datalist>`、`<keygen>`、`<output>`。不是所有的浏览器都支持 HTML5 新增的表单元素，但即使浏览器不支持该表单元素，也仍然可以显示为常规的表单元素。

5.3.1 `<datalist>` 元素

`<datalist>` 元素规定输入域的选项列表。`<datalist>` 属性规定 `form` 或 `input` 域应该拥有自动完成功能。当用户在自动完成域中开始输入时，浏览器应该在该域中显示填写的选项。Internet Explorer (IE) 9 之前的版本，Safari 都不支持 `<datalist>` 元素。

列表是通过 `<datalist>` 内的 `option` 元素创建的，如需把 `<datalist>` 绑定到输入域，请用输入域的 `list` 属性引用 `<datalist>` 的 `id`。我们通过一个实例来了解 `<datalist>` 元素的用法，代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>datalist 表单元素</title>
6 </head>
7 <body>
8   <form action="#">
9     <!-- list 属性的值要与 <datalist> 的 id 一致 -->
10    链接: <input type="text" list="url_list" />
11    <datalist id="url_list">
12      <!-- option 的 value 值必填, label 属性也会显示, 用于给当前的选项提示 -->
13      <option label="兄弟连IT教育" value="www.itxdl.cn" />
14      <option label="兄弟会" value="www.itxdh.cn" />
15      <option label="高洛峰主页" value="www.itxdl.cn" />
16      <option label="李明霞主页" value="www.limingxia.com" />
17      <option label="刘滔的主页" value="www.liutao1995.top" />
18    </datalist>
19    <input type="submit" value="提交" />
20  </form>
21 </script>
22 </body>
23 </html>
```



使用 `<datalist>` 元素的要点：需要为使用此 `list` 内容的 `<input>` 标签增加 `list` 属性，`list` 属性的值为 `<datalist>` 的 `id`；`<option>` 标签必须有 `value` 属性；`lable` 属性的内容也会显示出来，对当前的选项起提示作用。我们可以在文本框内单击下拉按钮展开所有选项，也可以通过输入选项内容展示符合要求的内容。效果如图 5.3 和图 5.4 所示。

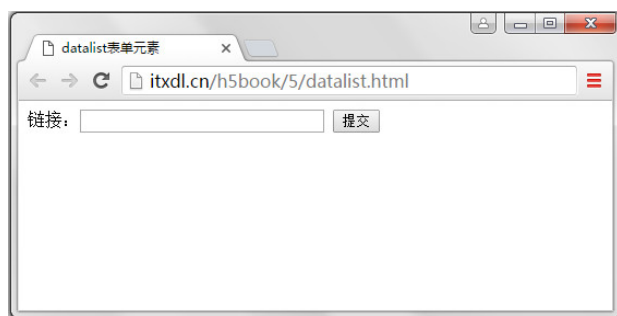


图 5.3 <datalist>元素（未输入时）

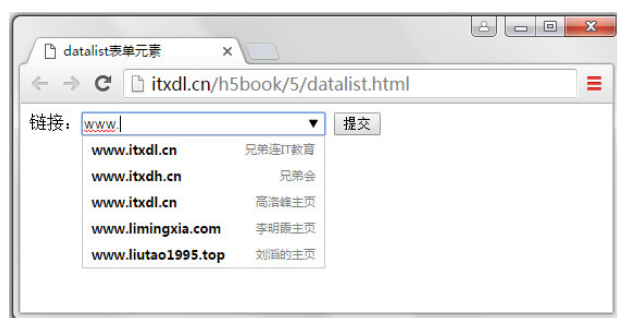


图 5.4 <datalist>元素（文本输入）

5.3.2 <keygen>元素

<keygen>元素的作用是提供一种验证用户的可靠方法。<keygen>元素规定用于表单的密钥对生成器字段。当提交表单时，会生成两个键，一个是私钥，另一个是公钥。私钥（Private Key）存储于客户端，公钥（Public Key）则被发送到服务器，公钥可用于之后验证用户的客户端证书（Client Certificate）。

这里我们通过一个 form 表单，讲解<keygen>元素的使用方法，最后通过表单提交到服务器，从而使服务器可以获取到加密后的内容。代码如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>keygen元素</title>
6     <style>
7     </style>
8 </head>
9 <body>
10    <form action="deal.php">
11        用户名: <input type="text" name="username" />
12        <!-- 将此文本框的内容加密 -->
13        加密内容: <keygen name="security" />
14        <input type="submit" value="提交" />

```





```
15     </form>
16     </script>
17 </body>
18 </html>
```

上述代码展示了一个加密的表单，单击“提交”按钮后，表单的数据和公钥都会提交到服务器，如图 5.5 所示。

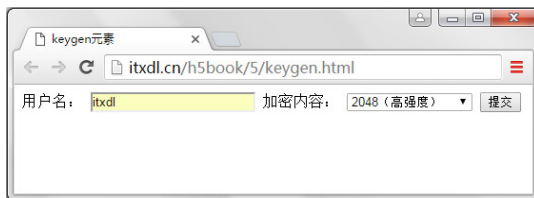


图 5.5 <keygen>元素

5.3.3 <output>元素

<output>元素用于不同类型的输出，比如计算或脚本输出。下面通过一道计算题来了解<output>元素标签，代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>output元素</title>
6     <style>
7     </style>
8 </head>
9 <body>
10     <form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
11         <input type="range" id="a" value="50">100
12         +<input type="number" id="b" value="50">
13         =<output name="x"></output>
14     </form>
15 </body>
16 </html>
```



上述代码阐述了一道加法计算题，目的是将前两个<input>标签内的值相加，输出到<output>标签内，效果如图 5.6 所示。

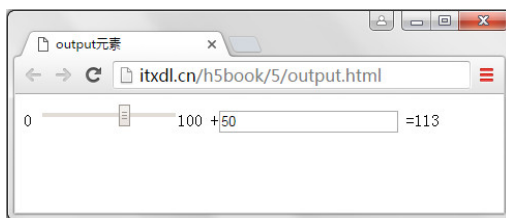


图 5.6 <output>元素

5.4 多样的输入类型

HTML5 拥有多个新的表单输入类型，这些新特性提供了更好的输入控制和验证。并不是所有的主浏览器都支持新的 `input` 类型，不过我们可以在所有的主浏览器中使用它们，即使不被支持，也仍然可以显示为常规的文本域。HTML5 `input` 类型如表 5.3 所示。

表 5.3 HTML5 input 类型

input 的类型	描 述
email	用于包含 E-mail 地址的输入域
url	用于包含 URL 地址的输入域
number	用于包含数值的输入域
range	用于包含一定范围内数字值的输入域，range 类型显示为滑动条
date picker	日期选择器，可选 date、month、week、time、datetime、datetime-local
search	用于搜索域
color	用于选取颜色

5.4.1 email

email 类型用于包含 E-mail 地址的输入域，在提交表单时，会自动验证 email 域的值，用法如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>email类型</title>
6   <style>
7     </style>
8 </head>
9 <body>
10  <p>验证邮箱格式，若邮箱格式错误会提示</p>
11  <p>标准邮箱地址：skygao@itxdl.cn</p>
12  <form action="#">
13    <input type="email" name="email" />
14    <input type="submit" value="验证">
15  </form>
16 </body>
17 </html>
```



上述代码验证了 email 输入框的邮箱格式，若出错会有提示，效果如图 5.7 所示。

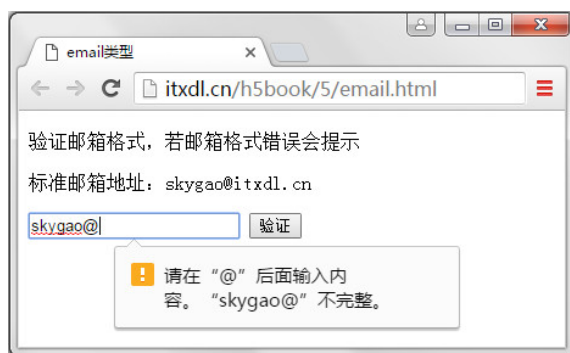


图 5.7 email 的输入类型

5.4.2 url

url 类型用于包含 URL 地址的输入域，在提交表单时，会自动验证 url 域的值，用法如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>url地址类型</title>
6   <style>
7   </style>
8 </head>
9 <body>
10  <p>验证url格式，若url格式错误会提示</p>
11  <p>标准url地址为: http://www.itxdl.cn</p>
12  <form action="#">
13    <input type="url" name="url" />
14    <input type="submit" value="验证">
15  </form>
16 </body>
17 </html>
```



上述代码验证了 url 地址输入框的输入格式，若出错会有提示，效果如图 5.8 所示。



图 5.8 url 的输入类型

5.4.3 number

number 类型用于包含数值的输入域，特别地，我们还可以对数值设置限定，用法如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>number类型</title>
6   <style>
7     </style>
8 </head>
9 <body>
10  <p>为number类型的input设置最小值为1，最大值为100</p>
11  <p>输入框可手动输入，或点击输入框右边的上下箭头使数值增减。</p>
12  <p>若输入的数值超出范围，页面会出现提示</p>
13  <form action="#">
14    <input type="number" name="number" min="1" max="100" />
15    <input type="submit" value="验证">
16  </form>
17 </body>
18 </html>

```



上述代码使用了 number 类型输入框，为该类型设置的数值范围为 1~100。当我们输入 101 时，数值超出了范围，页面会出现“值必须小于或等于 100”的提示，效果如图 5.9 所示。

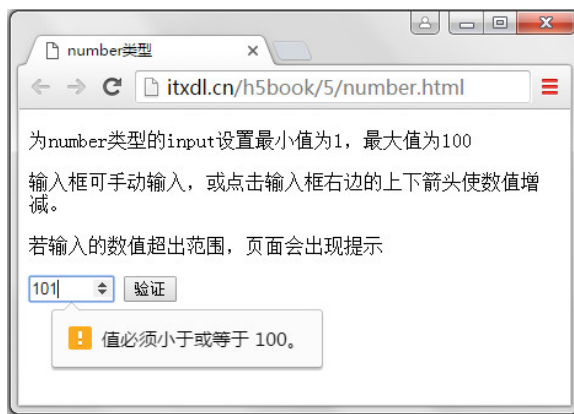


图 5.9 number 的输入类型

5.4.4 range

range 类型用于包含一定范围内数字值的输入域，跟 number 一样，我们还可以对数值设置限定，range 类型显示为滑动条，用法如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">

```




```
5     <title>range类型</title>
6 </head>
7 <body>
8     <p>为range类型的input设置最小值为1，最大值为100</p>
9     <p>滑动以改变输入域的值，点击提交弹出该值</p>
10    <form>
11        1<input id="range" type="range" name="range" min="1" max="100" />100
12        <input id="btn" type="submit" value="提交">
13    </form>
14    <script>
15        document.getElementById("btn").onclick=function(){
16            alert(document.getElementById('range').value);
17        }
18    </script>
19 </body>
20 </html>
```



上述代码使用了 range 类型输入框，为该类型设置的数值范围为 1~100。移动滑动条可以改变数值大小，提交后，弹出该值大小，效果如图 5.10 所示。



图 5.10 range 的输入类型

5.4.5 date picker

date picker 是日期选择器，HTML5 拥有多个可供选取日期和时间的新输入类型，具体选择如表 5.4 所示。

表 5.4 date picker 新输入类型

类 型	描 述
date	选取日、月、年
month	选取月、年
week	选取周和年

续表

类 型	描 述
time	选取时间、日、月、年（UTC 时间）
datetime-local	选取时间、日、月、年（本地时间）

下面我们以 `date` 为例，可以通过日期选择器从日历中选取一个日期，代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Date Picker日期选择器</title>
6 </head>
7 <body >
8     <p>这里的日期选择器的类型为date</p>
9     <p>我们可以从日历中选取一个日期</p>
10    <form>
11        <input type="date" name="date" />
12    </form>
13 </body>
14 </html>
```



上述代码使用了 `date`（日期）类型输入框，效果如图 5.11 所示。

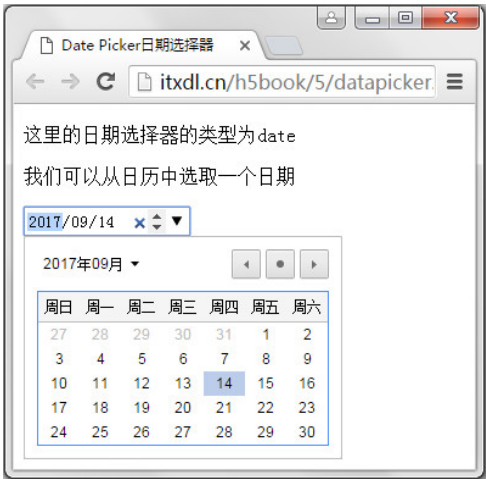


图 5.11 `date` 的输入类型

感兴趣的读者，可以使用其他类型的日期选择器，用法与 `date` 一致。

5.4.6 color

`color` 类型就是一个拾色器，用于规定颜色。该输入类型允许我们从拾色器中选取颜色，它的值为颜色的十六进制值。`color` 类型的输入框用法如下：



```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>color类型</title>
6   <style>
7     body{ margin-top:150px; }
8   </style>
9 </head>
10 <body>
11   <p>通过颜色选择器选择一种颜色，点击提交弹出该值</p>
12   <form>
13     <input id="color" type="color" name="color" />
14     <input id="btn" type="submit" value="提交">
15   </form>
16   <script>
17     document.getElementById("btn").onclick=function(){
18       alert(document.getElementById('color').value);
19     }
20   </script>
21 </body>
22 </html>
```



上述代码使用了 color 类型输入框，提交后出现拾色器以供我们选取特定的颜色，效果如图 5.12 所示。

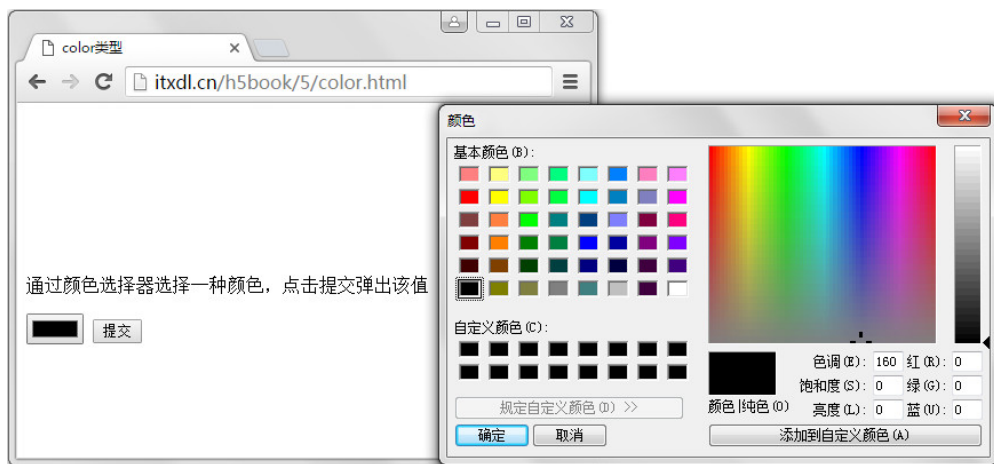


图 5.12 color 的输入类型

选取好颜色后，单击“提交”按钮，这时会弹出警示框显示该颜色的十六进制值，效果如图 5.13 所示。



图 5.13 color 的十六进制值

5.5 HTML5 新增的表单属性

HTML5 input 表单为<form>和<input>标签添加了几个新属性，如表 5.5 所示。

表 5.5 HTML5 新增的表单属性

属 性	描 述
autocomplete	规定 form 或 input 域应该拥有自动完成功能
autofocus	规定在页面加载时，域自动获得焦点
form	规定输入域所属的一个或多个表单
form overrides	表单重写属性，允许重写 form 元素的某些属性设定
height 和 width	规定用于 image 类型的<input>标签的图像高度和宽度
list	规定输入域的 datalist
min、max	用于为包含数字或日期的 input 类型规定限定（约束）
step	为输入域规定合法的数字间隔
multiple	规定<input>标签可以选择多个值
novalidate	规定在提交表单时不应该验证 form 或 input 域
pattern	该属性描述一个正则表达式，用于验证<input>标签的值
placeholder	该属性提供一种提示（hint），描述输入域所期待的值
required	规定必须在提交之前填写输入域（不能为空）



5.5.1 autocomplete属性

autocomplete 属性规定 form 或 input 域应该拥有自动完成功能，当用户在自动完成域中开始输入时，浏览器在该域中显示填写的选项。Autocomplete 属性适用于<form>标签，以及以下的<input>标签：text、search、url、email、password、date picker、range 及 color。autocomplete 属性有可能在 form 元素中是开启的，而在 input 元素中是关闭的。用法如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>autocomplete</title>
6   <style>
7     form{ width: 260px; height: 100px; margin: 40px auto; }
8     input{ margin: 5px auto; }
9     #btn{ display: block; margin: 10px auto; }
10  </style>
11 </head>
12 <body>
13   <!-- form域拥有自动完成功能，同时关闭password关闭自动完成功能 -->
14   <form action="#" autocomplete="on">
15     姓名: <input type="text" name="name" /><br>
16     密码: <input type="password" name="pwd" autocomplete="off" /><br>
17     邮箱: <input type="email" name="email" /><br>
18     主页: <input type="url" name="url" /><br>
19     <input id="btn" type="submit" value="提交" />
20   </form>
21 </body>
22 </html>
```



上述代码展示了一个 form 表单，我们为 form 域打开添加自动完成功能并关闭 password 密码域的自动完成功能。起初为表单填充我们想要的值，效果如图 5.14 所示。

图 5.14 autocomplete 属性（起初为表单填充）

自动填充功能提升了用户体验，为用户节省了时间。当用户完成了某网站大量的表单输

入，因为某一项不正确而返回表单页面后，若需要重新填写所有项则会影响用户体验，这时为表单的某些输入域设置自动填充功能可以为用户省去大量时间。图 5.15 所示是返回后的表单。

图 5.15 autocomplete 属性（自动填充）

因为我们为 form 开启了自动填充功能，所以表单中的文本域都被填充为提交前的数据，而密码框关闭了该功能，所以密码的内容重置为空。

5.5.2 autofocus 属性

autofocus 属性规定在加载时，域自动获得焦点。autofocus 属性是一个 boolean 属性，适用于所有<input>标签的类型。用法如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>autofocus属性</title>
6   <style>
7     form{ width: 260px; height: 100px; margin: 40px auto; }
8     input{ margin: 5px auto; }
9     #btn{ display: block; margin: 10px auto; }
10  </style>
11 </head>
12 <body>
13   <!-- 在页面加载时，邮箱域自动地获得焦点 -->
14   <form action="#">
15     姓名: <input type="text" name="name" /><br>
16     密码: <input type="password" name="pwd" /><br>
17     邮箱: <input type="email" name="email" autofocus="on" /><br>
18     主页: <input type="url" name="url" /><br>
19     <input id="btn" type="submit" value="提交" />
20   </form>
21 </body>
22 </html>

```





上述代码展示了一个 form 表单，在页面加载时，邮箱位置自动获取焦点，效果如图 5.16 所示。

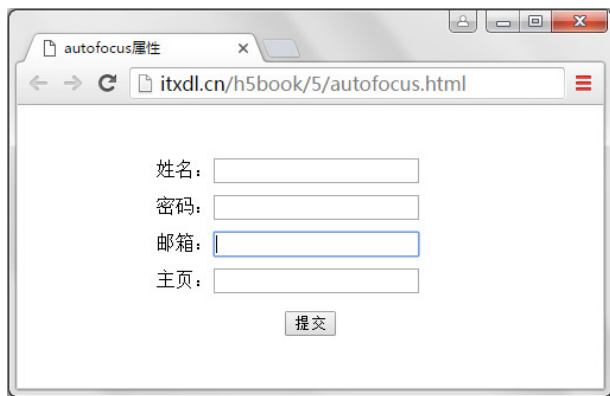


图 5.16 autofocus 属性

5.5.3 form 属性

form 属性规定输入域所属的一个或多个表单。form 属性适用于标签的类型，必须引用所属表单的 id。如需引用一个以上的表单，则使用空格分隔的列表。用法如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>autofocus属性</title>
6   <style>
7     form{ width: 260px; height: 100px; margin: 40px auto; }
8     input{ margin: 5px auto; }
9     #btn{ display: block; margin: 10px auto; }
10  </style>
11 </head>
12 <body>
13   <form id="user_form" action="#">
14     姓名: <input type="text" name="name" /><br>
15     密码: <input type="password" name="pwd" /><br>
16     邮箱: <input type="email" name="email" autofocus="on" /><br>
17     <input id="btn" type="submit" value="提交" />
18   </form>
19   <!-- 该input使用了form属性指向了上面的form表单，属于它的一部分 -->
20   主页: <input type="url" name="url" form="user_form" /><br>
21 </body>
22 </html>
```



上述代码包含了一个表单，其中先将姓名、密码、邮箱输入域包含在默认表单 user_form 中，之后通过 form 属性将 url 输入域包含在这个表单中。

5.5.4 form overrides表单重写属性

表单重写属性允许重写 form 元素的某些属性设定。表单重写属性适用于<input>标签的以下类型：submit 和 image。表单重写属性如表 5.6 所示。

表 5.6 表单重写属性

属 性	描 述
formaction	重写表单的 action 属性，用于描述表单提交的 URL 地址
formenctype	重写表单的 enctype 属性，用于描述表单提交到服务器的数据编码（只对 form 表单中 method="post" 表单）
formmethod	重写表单的 method 属性，定义了表单提交的方式，如 get、post
formnovalidate	重写表单的 novalidate 属性，描述了<input>元素在表单提交时无须验证
formtarget	重写表单的 target 属性，指定一个名称或一个关键字来指明表单提交数据接收后的展示，如_blank

下面我们创建一个表单，给予它默认的 action（提交地址）为“itxdl.php”，默认 method（提交方法）属性为“get”，target 属性为默认值“_self”，效果是在当前窗口打开。通过表单的重写属性来改变该表单原来设置的属性。代码如下：

```
13 <form action="itxdl.php" method="get">
14   姓名: <input type="text" name="name" /><br>
15   密码: <input type="password" name="pwd" /><br>
16   邮箱: <input type="email" name="email" /><br>
17   主页: <input type="url" name="url" /><br>
18   <input class="btn" formaction="itxdh.php" type="submit"
19     value="重新提交URL地址" />
20   <input class="btn" formnovalidate="true" type="submit"
21     value="不验证提交" />
22   <input class="btn" formtarget="_blank" type="submit"
23     value="重写target属性" />
24   <input class="btn" formmethod="post" type="submit"
25     value="重写method属性" />
26 </form>
```

5.5.5 height和width属性

height 和 width 属性规定用于 image 类型的<input>标签的图像高度和宽度。图像通常会同时指定高度和宽度属性。如果图像设置高度和宽度，则图像所需的空间在加载页面时会被保留。如果没有这些属性，浏览器不知道图像的大小，则不能预留适当的空间。图片在加载过程中会使页面布局效果改变。用法如下：



```
13 <form action="itxd1.php" method="get">
14   <!-- 为image类型的<input>标签设置高度和宽度 -->
15   <input type="image" src="./image/itxd1.png" width="193" height="258" />
16 </form>
```



在这段代码中，我们将 image 类型的<input>标签的高度 height 属性设置为 193，宽度 width 属性设置为 258，展示效果如图 5.17 所示。



图 5.17 为 image 类型的<input>标签设置 height 和 width

5.5.6 list属性

list 属性规定输入域的 datalist，datalist 是输入域的选项列表。list 属性适用的<input>标签类型有 text、search、url、email、date picker、number、range 和 color。详细用法可以参考本章“5.3.1 <datalist>元素”小节中的内容。

5.5.7 min、max和step属性

min（最小值）、max（最大值）和 step（步数间隔）属性用于为包含数字或日期的 input 类型规定限定（约束）。max 属性规定输入域所允许的最大值；min 属性规定输入域所允许的最小值；step 属性为输入域规定合法的数字间隔。min、max 和 step 属性适用的<input>标签类型有 date picker、number 及 range。下面我们给类型为 number 的<input>标签调整数字域，该域接受 0~100 的值，且 step（步数间隔）为 3。代码如下：

```
8 <form action="itxd1.php" method="get">
9   <p>请输入1~100内的值，且步进为3的数字</p>
10  <input type="number" min="1" max="100" step="3" />
11  <input class="btn" type="submit" value="提交">
12 </form>
```



在上面这段代码中，我们为 number 类型的标签设置最小值为 1，最大值为 100，且步长为 3。在输入域中，我们允许用户输入 3 的倍数的 1~100 的值，若值不符合该要求，则会得到图 5.18 所示的提示。



图 5.18 设置输入域的 min、max 和 step 属性

5.5.8 multiple属性

multiple 属性是一个 boolean 属性，它规定标签中可选择多个值。multiple 属性适用的标签类型有 email 和 file。用法如下：

```
8 <form action="itxdl.php" method="get">
9   <!-- 设置multiple属性可以允许我们选择多个文件 -->
10   选择文件: <input type="file" name="file" multiple="multiple" />
11 </form>
```



在上面这段代码中，我们为 file 类型的标签设置 multiple 属性，之后我们就可以选择多个文件，效果如图 5.19 所示。



图 5.19 为 file 文本域设置 multiple 属性

5.5.9 novalidate属性

novalidate 属性规定在提交表单时不应该验证 form 或 input 域。novalidate 属性适用的标签类型有 text、search、url、telephone、email、password、date picker、range 及 color。用法如下：



```
12 <body>
13 <!-- 为form表单设置novalidate属性为"true", 使表单不验证提交 -->
14 <!-- 这里的email和url都不被验证格式就能通过 -->
15 <form action="itxdl.php" method="get" novalidate="true">
16 姓名: <input type="text" name="name" /><br>
17 邮箱: <input type="email" name="email" /><br>
18 主页: <input type="url" name="url" /><br>
19 <input id="btn" type="submit" value="提交" />
20 </form>
21 </body>
```



在上面这段代码中,我们为表单设置 novalidate 属性,使表单不验证提交,表单中的 email 和 url 文本域不必通过验证。

5.5.10 pattern属性

pattern 属性规定用于验证 input 域的模式,即正则表达式。pattern 属性适用的<input>标签类型有 text、search、url、email 及 password。在下例中,我们为邮编的输入框设置 pattern 属性,只允许输入 6 位数字。用法如下:

```
8 <form action="itxdl.php">
9 <!-- 为该输入框设置pattern属性, 设定输入框内只能为6位数字 -->
10 邮编: <input type="text" name="postcode" pattern="[0-9]{6}" title="邮编号码为6位数字">
11 <input type="submit" value="提交邮编号码" />
12 </form>
```



在上面这段代码中,我们设置邮编号码只能为 6 位数字。若不符合这个正则表达式,浏览器就会提示“邮编号码为 6 位数字”,效果如图 5.20 所示。



图 5.20 为 text 文本域设置 pattern 属性

5.5.11 placeholder属性

placeholder 属性提供一种提示 (hint), 描述输入域所期待的值, 提示会在用户输入值前显示在输入域上。placeholder 属性适用的<input>标签类型有 text、search、url、email 及 password。用法如下:

```

13 <form action="itxdl.php" method="get">
14   <!-- 为<input>标签设置placeholder属性提供提示 -->
15   <input type="text" name="name" placeholder="请输入用户名" /><br>
16   <input type="password" name="pwd" placeholder="请输入密码" /><br>
17   <input type="email" name="email" placeholder="请输入邮箱" /><br>
18   <input type="url" name="url" placeholder="请输入主页链接" /><br>
19   <input id="btn" type="submit" value="提交" />
20 </form>

```



在上面这段代码中，我们为各个输入框都设置了 `placeholder` 属性，用来提示用户在这个输入框内需要填写的信息，效果如图 5.21 所示。

图 5.21 为文本域设置 `placeholder` 属性

5.5.12 required属性

`required` 属性规定在表单提交之前，用户必须填写该输入域，即不能为空。`required` 属性适用的 `<input>` 标签类型有 `text`、`search`、`url`、`email`、`password`、`date picker`、`number`、`checkbox`、`radio` 及 `file`。用法如下：

```

11 <form action="itxdl.php">
12   <!-- 为该输入框设置required属性，表示该域必填，不能为空 -->
13   姓名: <input type="text" name="name" required="required" />
14   <input type="submit" value="提交" />
15 </form>

```



在上面这段代码中，我们为“姓名”后的 `text` 输入框都设置了 `required` 属性，表示该输入框为必填项，若用户未输入任何值，则页面会出现如图 5.22 所示的提示。

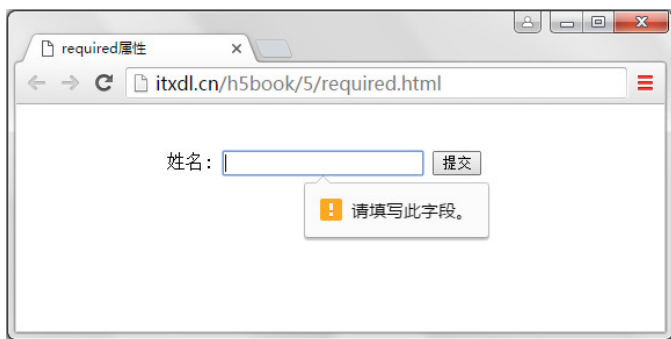


图 5.22 为文本域设置 required 属性

5.6 HTML5 表单提交综合实例

这里我们创建一个填写个人基本信息的表单，使用的表单元素有输入框、<datalist>选项列表、<textarea>文本框，通用的表单输入类型有 text、password、file、radio、checkbox、email、url、number、date picker，并为表单及表单元素设定相应的属性。代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>H5表单</title>
6   <style>
7     #box{ width: 960px; margin: 20px auto; }
8     #box h1{ font-size: 20px; text-align: center; }
9     #box form{ width:900px;border:1px dashed #333;padding:10px;height:550px; }
10    #box .pieces{ width: 380px; margin: 20px auto; }
11    #box .pieces span{ display: block; float: left; margin-bottom: 10px; }
12    #box .pieces span:nth-of-type(1){ width: 36%; text-indent: 2em; }
13    #box .pieces span:nth-of-type(2){ width: 64%; }
14    #box .pieces input{}
15    #btn{ margin: 0 auto; display: block; }
16  </style>
17 </head>
18 <body>
19   <div id="box">
20     <h1>请填写您的个人信息（*为必填项）</h1>
21     <form action="itxdl.php" autocomplete="true">
22       <div class="pieces">
23         <span>用户名</span>
24         <span>
25           <input type="text" name="username" placeholder="请输入用户名" required="true" />*
26         </span>
27       </div>
28       <div class="pieces">
29         <span>密码: </span>
```

```

30     <span>
31         <input type="password" name="pwd" placeholder="请输入密码" autocomplete="false"
32             required="true" />*
33     </span>
34 </div>
35 <div class="pieces">
36     <span>确认密码: </span>
37     <span>
38         <input type="password" name="repwd" placeholder="再次确认密码" autocomplete="false"
39             required="true" />*
40     </span>
41 </div>
42 <div class="pieces">
43     <span>邮箱: </span>
44     <span>
45         <input type="email" name="email" placeholder="请输入邮箱" required="true" />*
46     </span>
47 </div>
48 <div class="pieces">
49     <span>电话: </span>
50     <span>
51         <input type="text" name="phone" placeholder="请输入电话号码" required="true" />*
52     </span>
53 </div>
54 <div class="pieces">
55     <span>真实照片: </span>
56     <span>
57         <input type="file" name="photo" />
58     </span>
59 </div>
60 <div class="pieces">
61     <span>主页: </span>
62     <span>
63         <input type="url" name="url" placeholder="输入个人主页地址" />
64     </span>
65 </div>
66 <div class="pieces">
67     <span>性别: </span>
68     <span>
69         <label><input type="radio" name="gender" value="male" />男</label>
70         <label><input type="radio" name="gender" value="female" />女</label>
71     </span>
72 </div>
73 <div class="pieces">
74     <span>爱好: </span>
75     <span>
76         <label><input name="hobby" type="checkbox" value="" />跑步 </label>
77         <label><input name="hobby" type="checkbox" value="" />滑雪 </label>
78         <label><input name="hobby" type="checkbox" value="" />骑行 </label>
79         <label><input name="hobby" type="checkbox" value="" />其他 </label>
80     </span>
81 </div>
82 <div class="pieces">
83     <span>兄弟连校区: </span>
84     <span>
85         <input style="width: 220px" type="text" list="url_list" />
86         <datalist id="url_list">
87             <option label="北京市" value="北京校区" />

```




```
88     <option label="北京市" value="北京二校" />
89     <option label="上海市" value="上海校区" />
90     <option label="广州市" value="广州校区" />
91     <option label="辽宁省沈阳市" value="沈阳校区" />
92     <option label="河南省郑州市" value="郑州校区" />
93     <option label="山东省济南市" value="济南校区" />
94     <option label="四川省成都市" value="成都校区" />
95     <option label="浙江省杭州市" value="杭州校区" />
96     <option label="江苏省南京市" value="南京校区" />
97     <option label="广西省南宁市" value="南宁校区" />
98     <option label="天津市" value="天津校区" />
99     <option label="山西省太原市" value="太原校区" />
100 </datalist>
101 </span>
102 </div>
103 <div class="pieces">
104     <span>出生日期: </span>
105     <span>
106         <input type="date" name="birthday" required="true" />
107     </span>
108 </div>
109 <div class="pieces">
110     <span>地址: </span>
111     <span>
112         <input type="text" name="address" placeholder="请输入地址信息" />
113     </span>
114 </div>
115 <div class="pieces">
116     <span>邮编: </span>
117     <span>
118         <input type="text" name="postocode" pattern="[0-9]{6}" title="邮编由6位数字组成" />
119     </span>
120 </div>
121 <div class="pieces">
122     <span>个人简介</span>
123     <span>
124         <textarea name="info" placeholder="请填写个人简介" rows="3" cols="20"></textarea>
125     </span>
126 </div>
127 <div class="pieces">
128     <input id="btn" type="submit" value="提交" />
129 </div>
130 </form>
131 </div>
132 </body>
133 </html>
```



运行上面这段代码，表单在页面中的展示如图 5.23 所示。

请填写您的个人信息 (*为必填项)

用户名:

密码:

确认密码:

邮箱:

电话:

真实照片: 未选择任何文件

主页:

性别: ☐ 男 ☐ 女

爱好: ☐ 跑步 ☐ 滑雪 ☐ 骑行 ☐ 其他

兄弟连校区:

出生日期:

地址:

邮编:

个人简介:

图 5.23 HTML5 综合表单

本章小结

HTML 表单提交的方法有 `get` 和 `post` 两种, `get` 方法的作用是从指定的资源请求数据, `post` 方法的作用是向指定的资源提交要被处理的数据。HTML 表单一直都是 Web 的核心技术之一, 有了它, 我们才能在 Web 上进行各种各样的应用。HTML5 Form 新增了许多新控件及其 API, 方便我们做更复杂的应用, 而不用借助其他 JavaScript 框架。HTML5 新增表单元素有 `<datalist>`、`<keygen>` 和 `<output>`。 `<datalist>` 元素规定输入域的选项列表; `<keygen>` 元素的作用是提供一种验证用户的可靠方法; `<output>` 元素用于不同类型的输出, 比如计算或脚本输出。HTML5 拥有多个新的表单输入类型, 这些新特性提供了更好的输入控制和验证, 如 “email” 类型的文本框可以验证邮箱并提供提示。新增的表单属性用于对表单或表单文本域进行控制, 比如控制表单的自动完成、自动填充功能和文本域的提示 (hint)、正则匹配等功能。

本章习题

1. 以下哪项不是 HTML5 新增的 form 元素? ()
- A. `<datalist>` B. `<keygen>` C. `<output>` D. `<novalidate>`



2. 以下不是A. datetime B. file C. color D. range
3. 在 HTML5 中, onblur 和 onfocus 是 ()。
A. HTML 元素 B. 样式属性 C. 事件属性 D. 表单属性
4. 在 HTML5 中, 哪个属性用于规定输入字段是必填的? ()
A. required B. formvalidate C. validate D. placeholder
5. 以下哪种输入类型定义滑块控件? ()
A. search B. controls C. slider D. range
6. 在下列的 HTML 中, 哪一项可以产生复选框? ()
A. <input type="check"> B. <checkbox>
C. <input type="checkbox"> D. <check>
7. 关于 HTML5 的说法正确的是 ()。
A. HTML5 是在原有 HTML 上的升级版
B. HTML 可以不需要 DTD
C. 没有<!DOCTYPE html>, HTML5 可以正常工作
D. <output>是 HTML5 的新标签
8. 下列哪种输入类型用于定义周和年控件(无时区)? ()
A. date B. week C. year D. time
9. 在 url 类型的输入框中, 输入以下哪项 url 不会出现错误提示? ()
A. www.itxdl.cn B. https://itxdl.cn
C. https://www.itxdl.cn D. itxdl.cn
10. 生成类型为 range 的<input>标签, “min” 值为 0, “max” 值为 100, “step” 为 3。
输入以下哪一项的数字可以让此文本框通过验证? ()
A. 102 B. 10 C. 54 D. 100
11. 简述题: get 和 post 方法的区别有哪些? 何时使用 post 方法?
12. 简述题: HTML5 有哪些新增的表单元素?



本章习题及其答案



本章资源包



本章扩展知识

第6章

CSS3 揭秘



HTML 使用标签将内容放到网页上,也可使用元素和属性来控制简单的文档外观。如果希望更全面地控制 Web 页面的外观和布局,则需要使用层叠样式表(简称为 CSS)。CSS 规范的工作原理在于允许用户制定一些规则,描述文档中元素内容的表现形式,通过设置不同的规则控制页面中每个元素的外观,包括字体的颜色和大小、线的宽度和颜色、页面中各项之间的空白量,以便使页面看上去更令人感兴趣。CSS 和 HTML 一样,是所有网页制作技术的核心与基础,是为 HTML 制定样式的机制,能控制浏览器如何显示 HTML 中的每个元素及其内容。CSS 和 HTML 一起工作,用来弥补 HTML 对网页格式化功能的不足。既可以将 HTML 和 CSS 写在同一个文件中,也可以分开编写,两者都是纯文本文件,也都是通过浏览器解析的。本章所介绍的 CSS 语法只覆盖了本书程序实例中所涉及的内容。



本章二维码

本章二维码里面包括:

- (1) 本章的学习视频;
- (2) 本章所有实例演示结果;
- (3) 本章习题及其答案;
- (4) 本章资源包(包括本章所有代码)下载;
- (5) 本章的扩展知识。

6.1

CSS简介

CSS 是英文 Cascading Style Sheets (层叠样式表单) 的缩写,它是一种用来表现 HTML 或 XML 等文件样式的计算机语言,所以学习 CSS 之前应该先去了解 HTML。CSS 的作用是



定义网页的外观(如字体、背景、文本、位置、布局、边缘、列表及其他),其也可以和 JavaScript 等浏览器脚本语言合作做出许多动态的效果。

- 所谓样式表,是样式化 HTML 的一种方法。HTML 是文档的内容,而样式表是文档的表现,或者说外观。
- 所谓层叠,就是将一组样式放在一起层叠使用,控制某一个或多个 HTML 元素,按样式表中的属性依次显示。

一张样式表可以用于多个页面,甚至是整个站点,因此 CSS 具有良好的易用性和扩展性。从总体来说,使用 CSS 不仅能够弥补 HTML 对网页格式化功能的不足,如段落间距、行距、字体变化和大小等,还可以使用 CSS 动态更新页面格式、进行排版定位等。CSS 的特点如下:

- 控制页面中的每一个元素(精确定位)。
- 对 HTML 语言处理样式的最好补充。
- 把内容和格式处理相分离,减少工作量。

我们可以将 CSS 定义在 HTML 文档的每个标签中,或者以<style>标签嵌入在 HTML 文档中,也可以在外部附加文档作为外加文档。本例使用嵌入样式表,改变同一个 HTML 文档中 1 个<h1>和 4 个<p>标签的输出效果。使用文本编辑器打开一个扩展名为.html 的网页文件,将 3 个字符串分别编写在 HTML 的 1 个<h2>和两个<p>标签中,并在该文档中使用<style>标签嵌入 CSS 代码,控制这三个标签的显示效果。代码如下:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>一个使用css的简单实例</title>
6     <style>
7     h1{                                /* 为标题标记h1定义样式,使用多个样式层叠*/
8         font-family: "微软雅黑";      /* 设置标题字体类型为微软雅黑 */
9         font-size: 18px;               /* 设置标题的字体大小为18px */
10        text-align: center;            /* 设置标题的字体居中 */
11        color: #000;                  /* 设置标题字体颜色为黑色,rgb为000 */
12    }
13    p{                                /* 为段落标记p定义样式,使用多个样式层叠 */
14        height: 30px;                 /* 设置段落高度为30px */
15        line-height: 30px;           /* 设置段落行高为30px */
16        font-size: 15px;              /* 设置段落字体大小为15px */
17        text-align: center;           /* 设置段落字体居中 */
18        color: black;                 /* 设置段落颜色为黑色 */
19        background-color: #ddd;       /* 设置段落背景颜色为rgb为ddd的颜色 */
20        border: 1px solid red;        /* 设置段落边框大小为1px且颜色为红色实线*/
21    }                                  /* HTML中嵌入标记的结束标记 */
22    </style>
23 </head>
24 <body>
25     <h1>兄弟连IT教育</h1>
26     <p>隶属于易第优(北京)教育咨询股份有限公司</p>
```



```

27 <p>专注于IT技术培训，是国内专业的PHP/LAMP技术专业培训学校</p>
28 </body>
29 </html>

```

使用浏览器直接打开该文件，就可以看到浏览器对这个网页文件解释后的结果，如图 6.1 所示。



图 6.1 简单的 CSS 实例演示结果

在本例中，HTML 定义的网页结构使用 CSS 设置输出格式，可以将格式和结构分离。只要在 CSS 中改变某些属性，使用这个样式的所有 HTML 标签都会更新。

6.2 CSS规则的组成

CSS 和 HTML 一样，都是由 W3C 制定的标准，这里介绍的 CSS 的特性和功能来源于 CSS1 和 CSS2（CSS2 是根据 CSS1 扩展的）。W3C 也有新的版本更新，称为 CSS3。虽然浏览器已经准备开始实现 CSS3 某些方面的内容，但当前浏览器仍然无法支持其某些特性。一张样式表由样式规则组成，以告诉浏览器怎样去呈现一个文档。图 6.2 所示给出了关于 CSS 规则的一个示例。

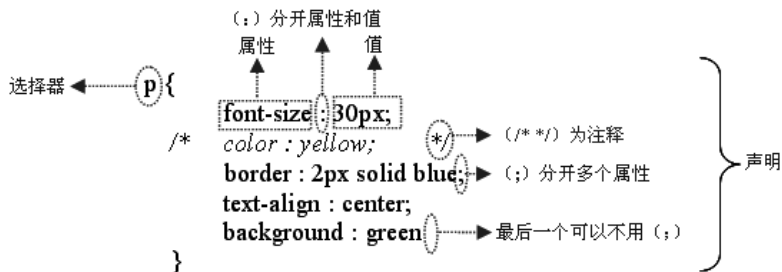


图 6.2 CSS 规则的组成



CSS 的规则主要由“选择器”和“声明”两部分组成。选择器指定声明应用于哪个或哪些元素（后面有详细讲解）。例如，任何 HTML 元素都可以是一个 CSS 的选择器，选择器仅是指向特别样式的元素。

```
p { text-indent: 3em } /* 当中的选择器是 p */
```

声明则是用于定义样式的元素，它用于设置 HTML 元素的样式。如果声明和选择器一起使用，就需要将声明的部分使用“{ }”组织在一起。声明中可以用多个样式属性，为通过该选择器找到的 HTML 元素叠加样式，每个样式元素都是由以冒号隔开的两部分组成的（属性:值）。属性是希望影响所选元素的特性，每个属性带一个值，共同描述选择器应该如何呈现。样式规则组成如下：

```
选择器 {属性 1:值 1;属性 2:值 2;属性 n:值 n;} /* 声明和选择器一起使用 */
```

属性和值之间使用冒号(:)连接，多个属性的复合样式声明之间应该用分号(;)隔开，最后一个属性的值后面可以不用分号。如果同一个样式属性出现两次以上，则使用后者。以下代码定义了 h1 和 h2 元素的颜色及字体大小属性。

```
2 <head>
3   <title>一个使用css的简单实例</title>
4   <style>
5     h1{ font-size: 18px; color: red; } /* 设置一级标题的字体大小为18px, 颜色为红色 */
6     h2{ font-size: 15px; color: blue; } /* 设置一级标题的字体大小为15px, 颜色为蓝色 */
7   </style>
8 </head>
```

如果直接在 HTML 标签中使用 style 属性声明样式，则不需要使用“{ }”，直接将多个层叠样式元素声明在 style 属性的双引号中即可。例如，直接将一个一级标题 h1 设置为加大、红色字体：

```
<h1 style="font-size:x-large;color:red"> 一级标题 </h1>      <!-- 在 HTML 标签中直接加样式 -->
```

6.2.1 CSS注释

样式表中的注释使用和 C 语言编程中一样的约定方法去指定，注释的内容会被浏览器忽略，可用于为样式表加注释及调试使用。CSS 注释的例子如下：

```
/* 这里的内容被注释，不能嵌套使用 */
```

6.2.2 长度单位

有很多样式属性的值都有长度单位，如 font-size、width、height、border-width 等。一个长度的值由可选的正号(+)或负号(-)、接着的一个数字、表示单位的两个字母组成。在

一个长度的值中间是没有空格的,如“1.3 em”就不是一个有效的长度的值,但“1.3em”就是有效的。一个为零的长度不需要两个字母的单位声明。无论是相对值还是绝对值长度,CSS 都支持。相对值单位确定一个相对于另一长度属性的长度,因为它能更好地适应不同的媒体,所以是首选的。CSS 中有很多专有的属性单位,也有很多能够用于大量属性的常规单位。

- em (font-size: 2em) 是一个大致与一个字符高度相同的单位。
- px (font-size: 12px) 是一个像素的单位。
- pt (font-size: 12pt) 是一个磅的单位。
- % (font-size: 80%) 是一个百分比。
- 其他单位还包括 pc (活字)、cm (厘米)、mm (毫米)、in (英寸)、dpi (输出分辨率) 和 rem (CSS3 新增的一个相对单位)。

另外,有的属性值还可以使用一个百分比,由可选的正号(+)或负号(-)、接着的一个数字、百分号(%)组成。在一个百分比值中间是没有空格的,百分比值是相对于其他数值的,同样用于定义每个属性。最常使用的百分比值是相对于元素的字体大小。

6.2.3 颜色单位和URL值

很多样式属性的值都有颜色单位,如 color、background-color、border-color 等。颜色值是一个关键字或一个 RGB 格式的数字。关键字通常有 16 个: aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white 和 yellow。RGB 颜色可以有以下 4 种形式,例子中指定同一种颜色:

- #rrggb (如#00cc00)。
- #rgb (如#0c0)。
- rgb(x,x,x), x 是一个 0~255 的整数 (如 rgb(0,204,0))。
- rgb(y%,y%,y%), y 是一个 0.0~100.0 的整数 (如 rgb(0%,80%,0%))。

另外,指定背景图片还需要使用一个 URL 值。URL 的格式为: url(addr), 其中 addr 是一个文件地址。URL 可以选择用单引号(')或者双引号("),也可以不加引号,并且在 URL 之前或之后可以包含空格。在 URL 中的括号、逗号、空格、单引号或双引号必须避开反斜杠。不完整的 URL 被理解为样式表的源代码,而不是 HTML 源代码。例如:

```
body { background: url(xsphp.gif) }           /* 不用引号 */
body { background: url(http://www.lampbrother.net/xsphp.gif) } /* 绝对 URL */
body { background: url('xsphp.gif') }        /* 使用单引号 */
body { background: url("xsphp.gif") }        /* 使用双引号 */
```



6.3 在HTML文档中放置CSS的几种方式

有很多种方法将样式表加入到 HTML 中，每种方法都有其优点和缺点。新的 HTML 元素和属性已被加入，以允许样式表与 HTML 文档更简易地组合起来。将样式表加入到 HTML 中的常用方法有内联样式表、嵌入一张样式表或链接到一张外部的样式表。

6.3.1 内联样式表

样式可以使用 style 属性内联，该属性可以应用于任意 body 元素（包括 body 本身），除了 basfont、param 和 script 元素。这个属性将任意数量的 CSS 声明当作自己的值，而每个声明用分号隔开，如下所示：

```
<p style="color: red; font-family: '微软雅黑'"> 此段落的样式是红色的“微软雅黑”字体</p>
```

内联样式表比其他方法更加灵活，但需要和展示的内容混合在一起，这样会失去样式表的一些优点。例如在本例中，如果有多个段落<p>标签都需要输出相同的样式，则在每个<p>标签中都需要使用 style 属性声明相同的样式，不仅需要的代码量比较大，而且不利于代码更新。

6.3.2 嵌入一张样式表

一张样式表可以使用<style>元素嵌入到 HTML 文档中使用，<style>元素需要放在 HTML 文档的 head 部分，代码如下：

```
2 <head>
3   <title>css样式表内嵌</title>
4   <style type="text/css">
5       body{ background-color:#fff; }
6       h1{ font-size: 18px; }
7       p{ color:#000; text-indent: 1em; }
8   </style>
9 </head>
```

其中，<style>和</style>标签之间是样式的内容（不要在这个标签中写 HTML 语句），type 属性表示使用的是文本中层叠样式表书写的代码。“{ }”前面是样式的选择器，“{ }”中是声明的样式属性。嵌入样式表对整个 HTML 文档都有效，可在一个 HTML 文档具有独一无二的样式时使用。

6.3.3 链接到一张外部的样式表

如果多个文档都使用同一张样式表，那么外部样式表会更适用。一张外部的样式表可以通过 HTML 的 link 元素链接到 HTML 文档中。<link/>标签放置在文档的 head 部分，可以通过多个<link/>标签链接多个样式文件到同一个 HTML 文档中，如下所示：

```
<link rel="StyleSheet" href="style.css" type="text/css" />  <!-- 在 HTML 中链接一个外部样式文件 -->
```

可选的 type 属性用于指定媒体类型，允许浏览器忽略它们不支持的样式表类型。rel 属性用于定义链接的文件和 HTML 文档之间的关系，该属性的值 StyleSheet 指定一个固定或首选的样式。而 href 属性则用来指定样式文件的位置，可以是相对 URL，也可以是绝对 URL。外部样式表文件要以扩展名.css 命名，并且在样式表文件中不能含有任何像<head>或<style>这样的 HTML 标签，样式表仅仅由样式规则或声明组成，如下所示：

```
p { margin: 2em } /* style.css 文件中的样式代码 */
```

在样式文件 style.css 中，一个单独由本例一条样式语法规则组成的文件，就可以用作外部样式表。当样式被应用到很多网页时，一张外部样式表是理想的。开发者使用外部样式表可以改变整个网站的外观，而且仅仅通过改变一个文件就可以实现。同样，大多数浏览器会在缓冲区保存外部样式表，从而避免在展示网页时发生延迟。

注意：如果同时使用内联样式表和嵌入一张样式表，并设置了相同属性，则内联样式表的优先级高；而同时使用嵌入一张样式表和链接到一张外部的样式表并设置相同属性时，优先级由出现的先后顺序决定。

6.4 CSS 普通选择器

HTML 有标签，而 CSS 有选择器。选择器就是赋予内部或者外部样式表的名字，当找到一个或多个 HTML 元素后，再通过声明属性加样式。常用的样式选择器包括：HTML 选择器、类选择器、id 选择器、关联选择器、组合选择器、伪类和伪元素。

6.4.1 HTML 选择器

HTML 选择器，即 HTML 的标签，用来改变一个指定标签的样式。任何 HTML 元素都可以是一个 CSS 的选择器，用于找到和选择器同名的 HTML 元素。如下所示：

```
5 h1{ font-size: 18px;line-height: 30px; } /* 这里的选择器是h1 */
6 p{ color:#000; text-indent: 1em; } /* 这里的选择器是p */
```




6.4.2 类选择器

同一个选择器能有不同的类（class），因而允许同一个元素有不同的样式。例如，开发者也许希望交替显示段落的背景颜色，代码如下：

```
5 p.dark_row{ background-color: #EAEAEA; } /* 定义<p>元素类名为dark_row的背景颜色 */
6 p.light_row{ background-color: #F8F8F8; } /* 定义<p>元素类名为light_row的背景颜色 */
```

这些类可以在 HTML 的<p>标签中使用 class 属性引用，每个<p>元素指定一个类名，代码如下：

```
10 <p class="dark_row">段落1, dark_row</p> <!-- 通过class属性指定段落使用dark_row类的样式 -->
11 <p class="light_row">段落2, light_row</p> <!-- 通过class属性指定段落使用light_row类的样式 -->
12 <p class="dark_row">段落3, dark_row</p> <!-- 通过class属性指定段落使用dark_row类的样式 -->
13 <p class="light_row">段落4, light_row</p> <!-- 通过class属性指定段落使用light_row类的样式 -->
```

以上例子建立了两个类：dark-row 和 light-row，供 HTML 的<p>元素使用，并通过 class 属性在 HTML 中指明元素使用的类。类的声明也可以不需要相关的元素，只要定义类选择器时不加点（.）前面的 HTML 标签即可，这样这个类就可以用于任何元素，代码如下：

```
.note { font-size: small } /* 为 note 的类可以被用于任何元素 */
```

类名是自定义的，一个良好的习惯是在命名类的时候，根据它们的功能而不是根据它们的外观命名。上述例子中的 note 类也可以命名为 small，但如果开发者决定改变这个类的样式，使得它不再是小字体，那么这个名称就变得毫无意义了。

另外，一个 HTML 元素可以同时使用多个类选择器，同样使用 class 属性指定多个类名，但多个类名之间要使用空格分开，代码如下：

```
<p class="one two three">第一段</p> /* 为 p 元素指定了 one、two 和 three 三个类 */
```

6.4.3 id选择器

在 HTML 页面中，id 属性指定了某个单一元素，id 选择器用来对这个单一元素定义单独的样式。id 选择器的应用和类选择器类似，只要把 class 换成 id 即可。它们的不同之处在于，id 用在唯一的元素上，而 class 则用在不止一个元素上。定义 id 选择器要在 id 名称前加上一个“#”符号，这和类选择器相同。例如，id 选择器可以指定如下：

```
#main { text-indent: 3em } /* 在 id 名称 main 前加上一个“#”符号 */
```

在下面的例子中，使用 id 属性匹配 id="main"的段落标签<p>：

```
<p id="main">文本缩进 3em</p> <!-- 在 HTML 的<p>标签中指定 id 属性值为 main -->
```

6.4.4 关联选择器

关联选择器只不过是一个用空格隔开的两个或更多的单一选择器组成的字符串。这些选择器可以指定一般属性，而且因为层叠顺序的规则，它们的优先权比单一的选择器大。这种方式只对第一个元素中关联的第二个元素定义（只要具有关联关系即可，关系的元素中间可以有更多其他 HTML 元素），对单独的第一个元素或第二个元素无定义。代码如下：

```
table a { color: red; } /* 只有在表格<table>内的链接<a>改变了样式 */
```

本例定义了表格<table>内的链接<a>改变了样式，文字颜色为红色，而表格外链接的文字颜色没有改变。

6.4.5 组合选择器

为了减少样式表的重复声明，组合选择器声明是允许的，只要用英文逗号（,）隔开选择器即可。例如，文档中所有的标题可以通过组合给出相同的声明，代码如下：

```
h1, h2, h3, h4, h5, h6 { color: red; font-family: sans-serif; } /* 使用组合选择器修饰标题 */
```

6.4.6 伪元素选择器

伪元素选择器是指对同一个 HTML 元素不同状态的一种定义方式。例如，对于<a>标签的正常状态、访问状态、选中状态、光标移到超链接文本上的状态，就可以使用伪元素选择器来定义。其语法结构如下：

```
HTML 标签:伪元素{属性:值;} /* 伪元素选择器的语法结构 */
```

每个伪元素都以英文的“:”开头，后面的伪元素名称根据作用不同有各自固定的写法。而冒号前面需要指定使用伪元素的 HTML 标签，目前只有<a>或<p>标签可用。指定超链接元素以不同的方式显示链接（links）、光标移动到超链接上（hover links）、已访问链接（visited links）和可激活链接（active links）时，定位伪元素可给出 link、hover、visited 或 active。一个已访问的链接可以定义为不同的颜色显示，甚至是不同的字体大小和风格。代码如下：

```
1 a:link {color:red; } /* 超链接没有任何动作前的状态 */
2 a:hover {color:yellow; font-size:125%; } /* 光标移动到超链接上的状态 */
3 a:active {color:blue; font-size:125%; } /* 选中超链接时的状态 */
4 a:visited {color:green; font-size:85%; } /* 访问过的超链接的状态 */
```

本例的效果是使当前链接在访问状态下，以不同颜色、更大的字号显示。而当网页的已访问链接被重选时，又以不同颜色、更小的字号显示。



注意：多个 CSS 选择器同时作用在同一个 HTML 元素上时，声明不同的属性具有继承的关系。如果声明的是相同属性，则以优先级高的选择器为主。主要选择器的优先级关系如下：

id 选择器 > 类选择器 > HTML 选择器 → 从左向右 CSS 选择器的优先级递减

最后还有一个比较特殊的标志——**!important**（权重），它没有属性值，但它的优先级是最高的。**!important** 属性是用来使用 IE 6、IE 7、IE 8、火狐（Firefox）、Google 等浏览器做兼容的，然而 IE 6 不能执行 **!important**。当 **!important** 规则被应用在一个样式声明中时，该样式会覆盖 CSS 中的其他声明，无论它处于声明列表中的哪个位置。尽管如此，**!important** 规则还是与优先级毫无关系，因为它改变了样式表的级联规则，让代码变得难以调试。笔者不建议使用 **!important** 属性，因为后期不利于维护。

6.5 CSS 常见的样式属性和值

CSS 中的样式属性比较多，经常使用的属性可以分为这么几类：字体、颜色、背景、文本、边框、列表，以及其他一些样式属性。每个类中的属性都可以单独使用；如果同一个类中多个属性一起使用，还可以将它们整合为一行解决。

6.5.1 字体属性

通过字体属性可以设置字体的族科，改变字体的大小和风格，也可以调整字体加粗，以及让字体变形等。修饰字体的所有属性、值及描述如表 6.1 所示。

表 6.1 CSS 中修饰字体的属性

属 性	描 述	属 性 值
font-family	字体族科	任意字体族科名称都可以使用，如 Times、serif 等，而且可以使用多个族科的赋值，中间用逗号分隔，以防止选择不存在的字体族科
font-size	字体大小	可以使用绝对大小、相对大小、长度或百分比
font-style	字体风格	normal（普通）、italic（斜体）或 oblique（倾斜）
font-weight	字体加粗	normal、bold、bolder 或 lighter 等
font-variant	字体变形	normal（普通）或 small-caps（小型大写字母）

分别使用表 6.1 中字体类的每个样式属性，指定 HTML 的段落元素 p 中的字体为 bold

(粗体)、italic (斜体)、Times 或 serif 字体、12 像素大小。代码如下：

```
5 p{ /* 选择器为HTML的段落元素p */
6     font-family: "楷体"; /* 设定字体族科为楷体 */
7     font-size: 12px; /* 设定字体大小为12px */
8     font-weight: bold; /* 设定字体加粗bold */
9     font-style: italic; /* 设定字体风格为斜体 */
10 }
```

本例中使用字体类中的多个属性设定段落中字体的样式，我们可以将其简化为使用一行代码解决。通过字体类中的 font 属性就可以做到，语法格式如下：

```
font: [<字体风格> || <字体变形> || <字体加粗> ]? <字体大小> [ / <行高> ]? <字体族科>
```

字体属性 font 用作不同字体属性的略写，特别是行高。允许值都是可选的，如果有多个属性值，则它们之间使用空格分开。例如，可以将上例代码改写为：

```
p { font: italic bold 12px/14px Times, serif } /* 所有字体属性一行解决 */
```

指定该段为 bold（粗体）和 italic（斜体），Times 或 serif 字体，12 像素大小，行高为 14 像素，和前面分别设定字体属性及值效果相同。

6.5.2 颜色属性

颜色属性允许网页制作者指定一个元素的颜色，在 CSS 中可以使用 color 属性设定文本的颜色。为了避免与用户样式表之间的冲突，背景和颜色属性应该始终一起指定。一些颜色规则的例子如下：

```
5 h1{ color: black; } /* 设置一级标题的字体颜色为黑色 */
6 h2{ color: #000; } /* 设置二级标题的字体颜色为黑色 */
7 h3{ color: #000000; } /* 设置三级标题的字体颜色为黑色 */
```

6.5.3 背景属性

大多数 HTML 元素都允许控制背景，包括背景颜色、背景图像、背景重复、背景附件、背景位置等属性。常见的控制背景属性、值及描述如表 6.2 所示。

表 6.2 CSS 中常见的控制背景的属性

属 性	描 述	属 性 值
background-color	背景颜色	值和 color 属性值设定方式相同，或使用 transparent（透明）值
background-image	背景图像	图片 URL 或 none（无）
background-repeat	背景重复	repeat、repeat-x、repeat-y、no-repeat
background-attachment	背景附件	scroll（滚动）或 fixed（固定）



续表

属 性	描 述	属 性 值
background-position	背景位置	横向的关键字(left, center, right), 纵向的关键字(top, center, bottom), 百分比和长度也可用来安排背景图像的位置
background-size	背景图片的尺寸	CSS3 的属性, 使用 (width,height) 来规定背景图片的宽和高
background-origin	背景图片的定位区域	CSS3 的属性, padding-box、border-box、content-box, 默认值是 padding-box

除了使用表 6.2 中提供的背景属性控制 HTML 元素的背景样式, 也可以将其简化为使用一行代码解决。通过背景类中的 background 属性实现, 语法规则如下:

```
background: <背景颜色> || <背景图像> || <背景重复> || <背景附件> || <背景位置>
```

背景属性 background 是一个更明确的背景, 是关系属性的略写。以下是一些背景的声明示例:

```
5 body{ background: #ddd url("./img/itxd1.jpg") no-repeat;  
6     background-size: 500px auto; background-position: center center; }  
7 h1{ background: #fff; }  
8 h2{ background: red; }  
9 h3{ background: #d3d3d3 url("./img/itxd1.jpg") no-repeat bottom right;  
10    background-size: 30px 40px; }
```

当一个值未被指定时, 将接受其初始值。例如, 在上述的前三条规则中, 背景的横向和纵向位置属性将被分别设置为 0%。

6.5.4 文本属性

CSS 文本属性主要包括字母间隔、文字间隔、文字修饰、文本排列、文本缩进、行高, 以及文字大小写等, 如表 6.3 所示。

表 6.3 CSS 中常见的控制文本的属性

属 性	描 述	属 性 值
letter-spacing	字母间隔	该值必须符合长度格式, 允许使用负值
word-spacing	文字间隔	该值必须符合长度格式, 允许使用负值
text-decoration	文字修饰	underline (下划线)、overline (上划线)、line-through (删除线)、blink (闪烁), 或默认使用无
text-align	文本排列	left、right、center 或 justify
text-indent	文本缩进	该值必须是一个长度或一个百分比。若为百分比, 则视上级元素的宽度而定

续表

属 性	描 述	属 性 值
line-height	行高	可以接受一个控制文本基线之间的间隔的值。当值为数字时，行高由元素字体大小的量与该数字相乘所得。百分比的值相对于元素字体的大小而定。不允许使用负值

6.5.5 边框属性

边框属性用于设置一个元素的边框风格、边框宽度、边框颜色，可以同时设置 4 条边的边框，也可对上边框、右边框、下边框和左边框单独设置。分别介绍如下。

1. 边框风格属性

可以通过边框风格属性 `border-style` 设定 4 条边框的风格，该属性用于设置一个元素边框的样式，而且必须用于指定可见的边框。可以使用 1~4 个关键字，如果 4 个值都给出了，则它们分别应用于上边框、右边框、下边框和左边框的样式。如果只给出一个值，则其将被运用到各边框上。如果给出了 2 个或 3 个值，则省略了的值与对边相等。也可以分别使用 `border-top-style`、`border-bottom-style`、`border-left-style` 和 `border-right-style` 属性单独设置各边框的风格，它们可以使用的属性值、解释和效果如图 6.3 所示。

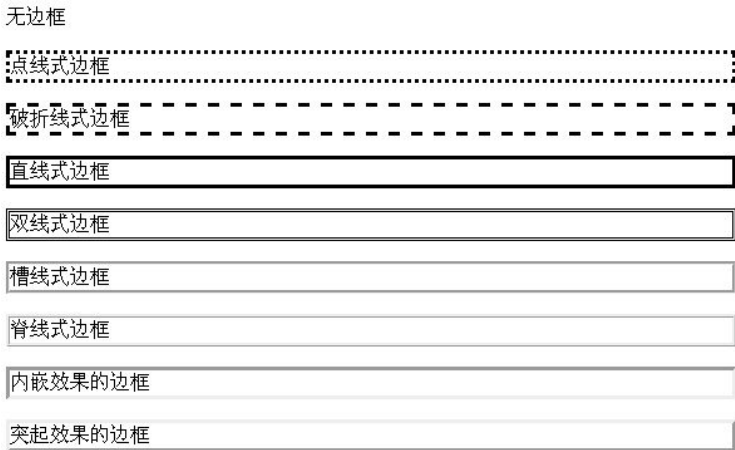


图 6.3 边框风格的属性值、解释和效果图

例句如下：

```
6 h1{ border-style: solid; } /* 设置一级标题h1的4个边框都为直线式边框 */
7 p{ border-style: dotted double; } /* 设置段落上下边框为虚线，左右边框为双线*/
```

2. 边框宽度属性

可以通过边框宽度属性 `border-width` 设定各边框的宽度，该属性用 1~4 个值来设置元



素的边界，值是一个关键字或长度，不允许使用负值长度。如果 4 个值都给出了，则它们分别应用于上边框、右边框、下边框和左边框的样式。如果只给出一个值，则其将被运用到各边框上。如果给出了 2 个或 3 个值，则省略了的值与对边相等。这个属性是上边框宽度、右边框宽度、下边框宽度和左边框宽度属性的略写。也可以分别使用 `border-top-width`、`border-bottom-width`、`border-left-width` 和 `border-right-width` 属性单独设置各边框的宽度。除了可以使用长度单位定值，还可以用 `medium`（默认值）、`thin`（比 `medium` 细）或 `thick`（比 `medium` 粗）值。例句如下：

```
6 p{
7   border-style: solid;           /* 设置段落的4个边框都为直线 */
8   border-left-width: 15px;      /* 设置段落左边框的宽度为15px */
9 }
```

3. 边框颜色属性

可以通过边框颜色属性 `border-color` 设定各边框的颜色，可以使用 1~4 个关键字。如果 4 个值都给出了，则它们分别应用于上边框、右边框、下边框和左边框的样式。如果只给出一个值，则其将被运用到各边框上。如果给出了 2 个或 3 个值，则省略了的值与对边相等。例句如下：

```
6 p{
7   border-style: solid;           /*设置段落的4个边框为实线 */
8   border-color: #FF0000 #0000FF; /*设置段落上下边框为红色，左右边框为蓝色*/
9 }
```

4. 略写的边框属性

CSS 属性 `border` 是边框属性的一个快捷的综合写法，是用于设置一个元素边框的宽度、样式和颜色的略写，它包含 `border-width`、`border-style` 和 `border-color` 属性。例句如下：

```
p {border:5px solid gray;}           /* 设置段落元素的 4 个边框为直线式边框、5 像素宽、灰色 */
```

边框属性 `border` 只能设置 4 种边框，也只能给出一组边框的宽度和样式。为了给出一个元素的 4 种边框的不同的值，网页制作者必须使用一个或更多的属性，如上边框、右边框、下边框、左边框、边框颜色、边框宽度、边框样式、上边框宽度、右边框宽度、下边框宽度或左边框宽度。

6.5.6 鼠标光标属性

在网页中默认的鼠标指针只有两种，一种是普通的箭头，另一种是当移动到链接上时出现的“小手”。但现在越来越多的网页都开始使用 CSS 鼠标指针技术，当鼠标移动到链接上时，可以看到多种不同的效果。CSS 可以通过 `cursor` 属性实现鼠标形状的改变，其属性可以是默认的鼠标形状 `default`、小手形状 `hand`、指示某对象可被移动 `move`、交叉十字 `crosshair`、

文本选择器号 text、Windows 的沙漏形状 wait、带有问号的鼠标 help 及各个方向的箭头属性值。例句如下：

```
p {cursor: pointer;} /* 当鼠标放在此项修饰的段落元素上时，出现“小手”形状的鼠标 */
```

6.5.7 列表属性

默认列表样式比较简单，但可以使用 CSS 中有关的属性丰富列表的外观。例如，可以在文本行前面加实心圆、空心圆、实心方块，可以在有序列表中使用阿拉伯数字、大写或小写的罗马数字、大写或小写的英文字母，还可以定制列表符号。其属性如表 6.4 所示。

表 6.4 CSS 中常见的控制列表的属性

属 性	描 述
list-style-type	设定引导列表项的符号类型，可以设置多种符号类型，值为 disc、circle、square 等
list-style-image	使用图像作为定制列表符号
list-style-position	决定列表项目缩进的程度

虽然可以使用 list-style-type 属性设定丰富的列表符号类型，而且也可以使用 list-style-image 属性添加自定义的列表符号，但是这些方法对符号的位置控制能力不强。比较常用的方法是关闭列表项自身的符号，然后使用定制的符号图像作为背景添加在列表元素上。这样就可以使用 CSS 的背景图像定位属性，精确地控制自定义符号的对齐方式。

不同的浏览器对列表样式的解析也不一样，IE 和 Opera 浏览器使用左外空白边距控制列表的缩进，而 Firefox 和 Safari 浏览器则使用左内填充空白边距控制列表的缩进。因此，在使用列表样式时，首先要将列表的左外空白边距（margin）和左内填充空白边距（padding）设置为 0，去掉所有边距的缩进。完成这些工作的例句代码如下：

```
1 ul {                                     /* 为HTML列表元素ul设置样式 */
2     list-style-type:none;               /* 列表样式类型设置为none，去掉默认的符号 */
3     margin:0px;                        /* 设定列表去掉四周的外边距的缩进 */
4     padding:0px;                       /* 设定列表去掉四周的内边距的缩进 */
5 }
```

接下来就可以添加自定义的符号了。首先在列表项左边用空白填充或使用文本缩进，为符号图像留出所需的空間，然后将自定义的符号图像作为背景图像应用于列表项中。如果一个列表项中的内容跨越多行，不希望将符号图像放置在第一行的位置，就可以将垂直位置设置为 center 或 50%，让符号图像垂直居中。例句如下：

```
1 /* 使用背景图像添加自定义的列表符号 */
2 li {                                     /* 为HTML列表项元素li设置样式 */
3     padding-left:30px;                  /* 在列表项左边填充空白，为符号留出所需的空間 */
```




```
4  /* 设置背景为images下的tp.gif图像, 图像不重复, 左边距离为0, 使用center值设置垂直居中 */
5  background:url(images/tp.gif) no-repeat 0 center;
6 }
```

6.5.8 CSS综合实例

在 Web 页面中经常使用栏目显示分类内容。本例将结合使用 HTML 和 CSS 编写一个分类栏目模型, 用于演示前面介绍的 CSS 应用。通过使用独立的文件定义样式表, 并在 HTML 文档中使用<link>标签与其链接, 使 HTML 代码和 CSS 代码完全分离。在 HTML 文档中只负责输出栏目的内容, 而栏目的样式则完全由 CSS 控制。创建一个名为 list.html 的 HTML 文档文件, 代码如下:

```
1 <html>
2   <head>
3     <title>css属性应用综合实例一定义栏目区块</title>           <!-- 页面标题 -->
4     <link rel="StyleSheet" href="style.css" type="text/css">       <!-- 链接外部样式文件 -->
5   </head>
6   <body>
7     <div id="wrapper">                                           <!-- 定义栏目所在区块容器 -->
8       <div class="tit">                                           <!-- 定义栏目标题容器 -->
9         <h3><a href="#">栏目标题</a></h3>                       <!-- 定义栏目标题 -->
10      </div>                                                      <!-- 标题容器结束 -->
11      <div class="list">                                           <!-- 定义栏目列表区块容器 -->
12        <ul>                                                       <!-- 定义无序列表项 -->
13          <li><a href="#">第一个列表选项</a></li>
14          <li><a href="#">第二个列表选项</a></li>
15          <li><a href="#">第三个列表选项</a></li>
16          <li><a href="#">第四个列表选项</a></li>
17          <li><a href="#">第五个列表选项</a></li>
18          <li><a href="#">第六个列表选项</a></li>
19          <li><a href="#">第七个列表选项</a></li>
20          <li><a href="#">第八个列表选项</a></li>
21          <li><a href="#">第九个列表选项</a></li>
22          <li><a href="#">第十个列表选项</a></li>
23          <li><a href="#">第十一个列表选项</a></li>
24          <li><a href="#">第十二个列表选项</a></li>
25        </ul>                                                     <!-- 列表项结束 -->
26      </div>                                                       <!-- 栏目内容区块容器结束 -->
27    </div>                                                         <!-- 栏目区块容器结束 -->
28  </body>
29 </html>
30
31
```

在上面的 HTML 文件中输出一个分类栏目, 包括栏目标题、栏目内容区块及内容列表等。但没有定义栏目的显示格式, 而是链接一个外部样式表文件 style.css, 由这个文件中的 CSS 代码控制输出栏目的样式格式。代码如下:

```
1 /* HTML选择器, 为HTML的body元素添加样式 */
2 body {
3   background:white;           /* 设置整个网页文本内容背景为白色 */
4   font:12px Arial,宋体;      /* 设置网页文字12像素Arial或宋体字 */
5 }
6 /* 组合选择器 + 伪元素选择器, 设置网页中正常和访问过的超链接的样式 */
```

```

7 a:link, a:visited {
8     text-decoration : none; /* 设置链接中的文本取消默认的下划线 */
9     color:#888; /* 设置链接中的文本字体颜色 */
10 }
11 /* id 选择器, 定义整个栏目容器的id选择器 */
12 #wrapper {
13     width: 300px; /* 定义栏目区块的宽度为300像素 */
14     text-align: left; /* 定义栏目区块的所有文本内容居左对齐 */
15 }
16 /* class 选择器, 定义栏目容器中的标题区块的类选择器 */
17 .tit {
18     width:100%; /* 栏目标题宽度占上一层标记的100% */
19     height:24px; /* 设置栏目标题区块高度为24像素 */
20     background:url(titbg.gif); /* 设置栏目标题的背景图片为titbg.gif */
21 }
22 /* 关联选择器, 设置栏目标题区块中的h3标题 */
23 .tit h3 {
24     margin:0px; /* 消除h3标题中默认的4边的空白外边距 */
25     padding:0px; /* 消除h3标题中默认的4边的空白内边距 */
26     line-height:24px; /* 设置h3标题的行高和标题区块高度相同 */
27     font-size:12px; /* 设置h3标题的字体大小为12像素 */
28     text-indent:30px; /* 设置h3标题的文本缩进为30像素 */
29     background:url(titbg.gif) no-repeat 3% 50%; /* 使用背景在h3标题前添加一张符号图片 */
30 }
31 /* class 选择器, 定义栏目容器中的内容列表区块的类选择器 */
32 .list {
33     width:298px !important; /* 内容盒子在非IE中的宽度为298px */
34     width:300px; /* 内容盒子在IE中显示宽度为300px */
35     float:left; /* 设置内容列表区块为左漂浮 */
36     border:1px solid #D8D8D8; /* 设置内容列表区块显示直线边框 */
37     border-top:0px; /* 设置内容列表区块取消顶部的边框线 */
38 }
39 /* HTML选择器, 为HTML列表元素ul 设置样式 */
40 ul {
41     list-style-type:none; /* 设置为none, 去掉列表默认的符号 */
42     margin:0px; /* 设定列表去掉四周的外边距的缩进 */
43     padding:0px; /* 设定列表去掉四周的内边距的缩进 */
44 }
45 /* 关联选择器, 为HTML列表项元素li 设置样式 */
46 ul li {
47     float:left; /* 设置每个列表项区块为左漂浮 */
48     line-height:20px; /* 设置列表项的行高为20像素 */
49     width:45%; /* 设置列表项的宽度占上一层标记的45% */
50     margin:0px 5px; /* 上下外边距为0, 左右边距为5px */
51     background: url(sidebottom.gif) repeat-x 0 bottom; /* 使用背景图像添加列表符号下方的点线 */
52 }
53 /* 关联选择器, 为列表元素中的超链接定义样式 */
54 ul a {
55     padding-left: 12px; /* 设置链接中的文本左内添加12px的空白 */
56     background: url(bullet.gif) no-repeat 0 50%; /* 使用背景图像为链接添加自定义的符号 */
57 }
58 /* 关联选择器, 设置网页中超链接被访问时的样式 */
59 ul a:hover {
60     text-decoration:underline; /* 设置超链接在被访问时添加下划线 */
61     color:#ff0000; /* 文本颜色设置为红色 */

```

通过将样式文件 style.css 嵌入到 HTML 文件 list.html 中, 使 HTML 文档中定义的各个元素被 CSS 样式控制。我们的初衷是让 HTML 文件的代码变得更简洁和易于维护。直接访问 list.html 文件的输出结果如图 6.4 所示。

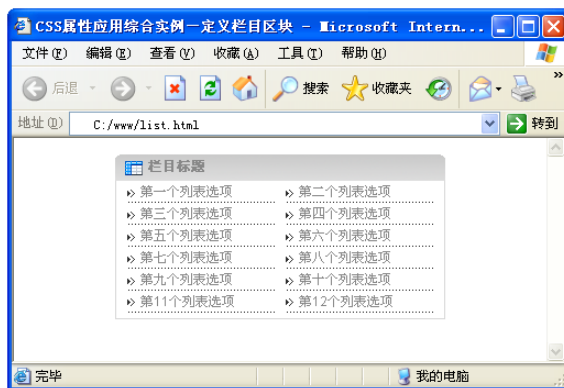


图 6.4 HTML 和 CSS 结合使用输出栏目内容

6.6 CSS3 概述

对于 Web 开发者来说，CSS3 不只是一门新奇的技术，更重要的是，这些全新概念的 Web 应用给开发人员带来了无限的可能性，也极大地提高了开发效率。我们不必再依赖图片或者 JavaScript 去完成圆角、多背景、用户自定义字体、3D 动画、渐变、盒阴影、文字阴影、透明度等提高 Web 设计质量的特色应用。

6.6.1 CSS3 在选择器上的支持

CSS3 在选择器上的丰富支持使得开发人员可以灵活地控制样式。通过选择器的使用，开发人员不再需要在编辑样式时使用多余的、没有任何语义的 `class` 属性，而可以直接将样式与元素绑定，利用属性选择器可以很容易地根据属性值的开头或结尾选择某个元素，利用兄弟选择器可以选择同级兄弟节点或紧邻下一个节点的元素，利用伪类选择器可以选择某一类元素，从而节省在网站或 Web 应用程序设计完成后又要修改样式所花费的时间。

6.6.2 CSS3 在样式上的支持

只要提起 CSS3 的特性是什么，回答最多的就是“圆角”。不错，“圆角”这个功能可以让前端布局节省大量的时间和精力去切图拼凑一个圆角。CSS3 还支持阴影（盒阴影和文本阴影）和渐变；可以自定义字体（使用 `@font-face`）；对于连续文本换行也新增了一些属性，这既解决了连续英文字符出现页面错位的问题，也不需要后端程序去截取这个连续字符；可以给边框添加背景，支持背景调整大小和背景透明处理。

6.6.3 CSS3 对于动画的支持

CSS3 支持的动画类型有：变换、过渡和动画。你可以对特定的属性设置 `transition` 和 `animation` 的值来实现动画效果。

6.6.4 在实际开发中该如何使用CSS3

首先应遵循一个优雅降级的原则，比如前面谈到的圆角，我们可以针对 Firefox 和 Safari 等支持圆角的浏览器应用 CSS 圆角，而那些不支持 CSS 圆角的浏览器则显示为直角。其次，对于不支持 CSS3 的浏览器，可以使用 JavaScript 脚本来实现。在向用户或企业推广新技术的同时，也要关注他们的目标与可行性，不能一味地追求技术。

6.7 CSS复杂选择器

要想使用 CSS 对 HTML 页面中的元素实现一对一、一对多或者多对一的控制，就需要用到 CSS 选择器。选择器是 CSS3 中的一个重要内容，使用它可以大幅度地提高开发人员书写或修改样式表的效率。在大型网站中，样式表中的代码可能会达到几千行，而当我们需要对样式进行修改时，并没有说明什么样式服务于什么元素，只是在元素中使用了 `class` 属性。而 `class` 属性本身没有语义，它纯粹是用来为 CSS 服务的，属于多余属性。CSS3 提倡使用选择器来对样式与元素进行直接绑定，这样一来，在样式中什么样式与什么元素相匹配就变得一目了然，修改起来也很方便。不仅如此，通过选择器我们还可以实现各种复杂的指定，同时也能大量减少样式表的代码书写量，使得最终书写出来的样式表变得简洁明了。

6.7.1 基本选择器

CSS 是一种用于屏幕上渲染 HTML 的语言，它主要是在相应的元素中应用样式来渲染相对应的元素，那么如何选择相应的元素就显得很重要了。这时可以应用选择器。选择器主要用来确定 HTML 的树形结构中的 DOM 元素节点。可以把 CSS 选择器分为基本选择器、属性选择器、伪类选择器等几个部分。CSS3 基本选择器如表 6.5 所示。

表 6.5 CSS3 基本选择器及实例

选 择 器	说 明
*	通用元素选择器，匹配任何元素



续表

选 择 器	说 明
E	标签选择器，匹配所有使用 E（所有 HTML 元素）的元素
.info	class 选择器，匹配所有 class 属性中包含 info 的元素
#footer	id 选择器，匹配所有 id 属性等于 footer 的元素
<pre>* { margin:0; padding:0; } /* 通用元素选择器，匹配任何元素 */ .info * { border:1px solid blue; } /* 选择某个元素下的所有元素 */ p { font-size:2em; } /* 标签选择器，匹配所有使用p标签的元素 */ .info { background:#ff0; } /* class选择器，匹配所有class属性中包含info的元素 */ p.info { background:#ff0; } /* class选择器，特定元素下的类属性 */ #info { background:#ff0; } /* id选择器，匹配所有id属性等于info的元素 */</pre>	

6.7.2 多元素的组合选择器

多个选择器可以组合在一起使用，其组合形式有多种，包括多元素选择器、后代元素选择器、子元素选择器、毗邻元素选择器、同级元素通用选择器 5 种类型，如表 6.6 所示。

表 6.6 CSS3 多元素的组合选择器及实例

选 择 器	说 明
E,F	多元素选择器，同时匹配所有 E 元素和 F 元素，E 和 F 之间用逗号分隔
E F	后代元素选择器，匹配所有属于 E 元素后代的 F 元素，E 和 F 之间用空格分隔
E > F	子元素选择器，匹配所有 E 元素的子元素 F（第一层后代）
E + F	毗邻元素选择器，匹配所有紧随 E 元素之后的同级元素 F
E ~ F	同级元素通用选择器，匹配任何在 E 元素之后的同级 F 元素
<pre>#nav,li { display:inline; } /* 多元素选择器，同时匹配所有id为nav元素或li元素 */ div p { color:#f00; } /* 后代元素选择器，匹配所有属于div元素后代的p元素 */ div > p { color:#f00; } /* 子元素选择器，应用于div元素的直接子元素p上 */ h1 + p { color:#f00; } /* 毗邻元素选择器，匹配所有紧随h1元素之后的同级元素p */ p ~ ul { background:#ff0; } /* 匹配任何在p元素之后的同级ul元素 */</pre>	

6.7.3 属性选择器

属性选择器可以根据元素的属性及属性值来选择元素。如果希望选择具有某个属性的元素，而不论属性值是什么，则可以使用简单属性选择器。在 CSS2.1 中就引入了一些属性选择器，如表 6.7 所示。

表 6.7 CSS2.1 属性选择器及实例

选 择 器	说 明
E[att]	匹配所有具有 att 属性的 E 元素，不考虑它的值（注意：E 在此处可以省略，比如 “[checked]”，以下同）
E[att=val]	匹配所有 att 属性值等于 “val” 的 E 元素
E[att~=val]	匹配所有 att 属性具有多个空格分隔的值，其中一个值等于 “val” 的 E 元素
E[att =val]	匹配所有 att 属性具有多个连字符分隔的值，其中一个值以 “val” 开头的 E 元素，主要用于 lang 属性，比如 “en”、“en-us”、“en-gb” 等
<pre>p[title] { color:#f00; } /* 匹配所有具有title属性的p元素，不考虑title的值 */ div[class=error] { color:#f00; } /* 匹配所有class属性等于"error"的div元素 */ td[headers~=coll] { color:#f00; } /* 匹配所有headers属性具有多个空格分隔的值，其中一个值等于"coll"的td元素 */ p[lang =en] { color:#f00; } /* 匹配所有lang属性具有多个连字符分隔的值，其中一个值以"en"开头的p元素 */ blockquote[class=quote][cite] { color:#f00; } /* 可以多个属性组合使用 */</pre>	

在 CSS3 中增加了一些属性选择器，使用 CSS3 的属性选择器，既可以只指定元素的某个属性，也可以同时指定元素的某个属性及其对应的属性值。CSS3 的属性选择器如表 6.8 所示。

表 6.8 CSS3 属性选择器及实例

选 择 器	说 明
E[att^="val"]	属性 att 的值以 “val” 开头的元素
E[att\$="val"]	属性 att 的值以 “val” 结尾的元素
E[att*="val"]	属性 att 的值包含 “val” 字符串的元素
<pre>.demo a[href^="http://"] { background:orange;color:green; } /* 属性href的值以"http://"开头的元素 */ .demo a[href\$=".png"] { background:orange;color:green; } /* 属性href的值以".png"结尾的元素 */ .demo a[title*="site"] { background:black;color:white; } /* 属性title的值包含"site"字符串的元素 */</pre>	

6.7.4 结构性伪类选择器

在学习结构性伪类选择器之前，先了解两个概念：伪类选择器和伪元素选择器。伪类选择器是 CSS 中已经定义好的选择器，不能随便命名。常用的伪类选择器是使用在 a 元素上的几种，如 a:link、a:visited、a:hover、a:active。而伪元素选择器并不是针对真正的元素使用的选择器，而是针对 CSS 中已经定义好的伪元素使用的选择器。CSS 中有 4 种伪元素选择器，即 first-line、first-letter、before、after。例如，p:first-line{color:#ff0000;}，li:after{content:url(dot.png)}。而在 CSS3 中引入的结构性伪类选择器的共同特征是允许开发者根据文档树中的结构来指定元素的样式，下面分类进行介绍，如表 6.9～表 6.14 所示。



表 6.9 4 个最基本的结构性伪类选择器 root、not、empty 和 target

选 择 器	说 明
E:root	将样式绑定到页面的根元素中。所谓根元素，是指位于文档树中最顶层结构的元素，在 HTML 页面中就是指包含整个页面的<html>部分
E:not	想对某个结构元素使用样式，但想排除这个结构元素下的子结构元素，就使用 not 样式
E:empty	指定当元素内容为空白时使用的样式
E:target	对页面中某个 target 元素指定样式，该样式只在用户单击了页面中的链接，并且跳转到 target 元素后生效
<pre>:root { background-color:yellow } /* 指定整个网页背景色为黄色 */ body *:not(h1) { background-color:yellow } /* 指定body元素背景为黄色，但排除h1 */ :empty { background-color:yellow } /* 指定某个元素内容为空白时，该元素背景为黄色（例如表格）*/ target { background-color:yellow } /* 链接的目标背景为黄色 */ /* 示例 <div id="text3">内容 </div> 点击时目标div背景为黄色 */</pre>	

表 6.10 选择器 first-child、last-child、nth-child、nth-last-child

选 择 器	说 明
E:first-child	对一个父元素中的第一个子元素指定样式
E:last-child	对一个父元素中的最后一个子元素指定样式
E:nth-child	对指定序号的子元素设置样式（正数） :nth-child(odd): 所有正数下来第偶数个子元素 :nth-child(even): 所有正数下来第奇数个子元素
E:nth-last-child	对指定序号的子元素设置样式（倒数） :nth-last-child(odd): 所有倒数上去第偶数个子元素 :nth-last-child(even): 所有倒数上去第奇数个子元素
<pre>p:first-child { color:yellow } /* 第一个p元素的样式 */ p:last-child { color:yellow } /* 倒数第一个p元素的样式 */ p:nth-child(2) { color:yellow } /* 第2个p元素的样式 */ p:nth-last-child(2){ color:yellow } /* 倒数第2个p元素的样式 */</pre>	

表 6.11 选择器 nth-of-type、nth-last-of-type

选 择 器	说 明
E:nth-of-type(n)	与:nth-child()作用类似，但是仅匹配使用同种标签的元素
E:nth-last-of-type(n)	与:nth-last-child()作用类似，但是仅匹配使用同种标签的元素
<pre>h2:nth-of-type(even) { color:yellow } /* h2元素的偶数行为黄色 */ h1:nth-of-type(odd) { color:green } /* h1元素的奇数行为绿色 */</pre>	

表 6.12 循环使用样式

选 择 器	说 明
E:nth-child(an+b)	a 表示每次循环中共包括几种样式；b 表示指定的样式在循环中所处的位置
<pre>li:nth-child(4n+1){background-color:yellow;} /*第一个li背景色为黄色，这样依次循环下去 */ li:nth-child(4n+2){background-color:bule;} /*第二个li背景色为蓝色..... */ li:nth-child(4n+3){background-color:red;} /*第三个li背景色为红色..... */ li:nth-child(4n+4){background-color:green;} /*第四个li背景色为绿色.....(4n+4可缩写为4n) */ /* 前面所讲的nth-child(odd) & nth-child(even) 可以用如下代码替代： */ li:nth-child(2n+1){} /* 所有正数下来的第奇数个子元素 */ li:nth-child(2n+2){} /* 第偶数..... */ li:nth-last-child(2n+1){} /* 所有倒数上去的第奇数个子元素 */ li:nth-last-child(2n+2){} /* 第偶数..... */</pre>	

表 6.13 选择器 only-child、only-of-type

选 择 器	说 明
E:only-child	匹配父元素下仅有的一个子元素 等同于: first-child:last-child 或:nth-child(1):nth-last-child(1)
E:only-of-type	匹配父元素下使用同种标签的唯一一个子元素 等同于: first-of-type:last-of-type 或:nth-of-type(1):nth-last-of-type(1)
<pre>li:nth-child(1):nth-last-child(1){ background-color:yellow} /* 可用only-child代替 */ li:only-child { background-color:yellow} /* 同上..... */ li:only-of-type{ background-color:yellow} /* 也可用only-of-type代替 */</pre>	

表 6.14 CSS3 中与用户界面有关的伪类选择器

选 择 器	说 明
E:enabled	匹配表单中激活的元素
E:disabled	匹配表单中禁用的元素
E:checked	匹配表单中被选中的 radio（单选按钮）或 checkbox（复选框）元素
E::selected	匹配用户当前选中的元素
<pre>input[type="text"]:disabled { background:#ddd; } /* 将表单中禁用的文本text元素背景设置成灰色 */</pre>	

nth-child 与 nth-of-type 的区别：

nth-child 和 nth-of-type 都是 CSS3 中的结构性伪类选择器，两者作用近似却又不完全一样，对它们不熟悉的人可能不是很容易区分。下面介绍两者的不同，以便于大家正确灵活地使用这两类选择器。

先看一个简单的实例，首先是 HTML 部分：



```
<div id="box">
  <p>我是段落 1</p>
  <p>我是段落 2</p>
</div>
```



两个选择器相对应的 CSS 代码如下：

```
#box p:nth-child(2) { color: red; }
```

```
#box p:nth-of-type(2) { color: red; }
```

在上面这个例子中，这两个选择器所实现的效果是一样的，第二个<p>标签的文字变成了红色，如图 6.5 所示。

尽管上面两个选择器最后实现的效果一样，但两个选择器之间存在差异是必然的。

对于 `nth-child` 选择器，意味着选择一个元素满足如下条件：

- (1) 这是一个段落元素。
- (2) 这是父标签的第二个子元素。



图 6.5 伪元素选择器实现效果 (1)

对于 `nth-of-type` 选择器，如果选择父标签的第二个段落子元素：

我们把上面的实例稍作修改，就可以看到这两个选择器之间的差异表现了。HTML 代码如下：

```
<div id="box">
  <h1>我是标题</h1>
  <p>我是段落 1</p>
  <p>我是段落 2</p>
</div>
```

同样，使用上面的 CSS 代码：

```
#box p:nth-child(2) { color: red; }
```

```
#box p:nth-of-type(2) { color: red; }
```



这时候两个选择器所渲染的结果就不一样了。

`p:nth-child(2)`没有达到原意，其渲染的结果不是第二个`<p>`标签文字变红，而是第一个`<p>`标签，也就是父标签的第二个子元素，效果如图 6.6 所示。



图 6.6 伪元素选择器实现效果（2）

`p:nth-of-type(2)`达到了我们想要的效果，其把希望渲染的第二个`<p>`标签染红了，效果如图 6.7 所示。



图 6.7 伪元素选择器实现效果（3）

对照上面两个选择器的语义，此处的效果表现差异不难理解。`nth-child`与`nth-of-type`的区别也可以看出。

6.8 CSS3 属性

通过 CSS 选择器找到元素后，使用 CSS 属性给找到的元素设置样式。尽管现在有些浏览器已经支持众多的 CSS3 属性，但作为初学者，最应该关注的是一些“主流”的属性，如 `border-radius`、`box-shadow`、`transform` 等。它们有良好的文档、极佳的测试效果，并且经常用到，将成为你设计网站的得力助手。学习 CSS3 的属性先从主流的属性开始，其他属性可



通过 CSS3 手册获得帮助。具体实例展示，读者可以在下一章大量学习。本节主要学习 CSS3 属性的用法及兼容性。

6.8.1 使用CSS3 属性前的准备

虽然目前主流的浏览器大多不支持 CSS3 的大多数属性，但还是鼓励读者在前端开发中学会并且运用这些 CSS3 属性，因为这是未来的发展趋势。关键是初学者首先要确定是否对各个浏览器之间的细微差别有所了解，但要牢记，网站设计不必看到所有浏览器的不同。在使用 CSS3 的一些高级特性时，需指定所有浏览器的前缀。CSS3 的前缀是浏览器生产商经常使用的一种方式，它暗示该 CSS 属性或规则尚未成为 W3C 标准的一部分。CSS3 前缀的浏览器属性规则如表 6.15 所示。

表 6.15 CSS3 前缀的浏览器属性规则

CSS3 前缀	代表浏览器
-webkit-	Chrome、Safari
-moz-	Firefox
-ms-	IE
-o-	Opera
<pre>-webkit-transform: rotate(-3deg); /* 为Chrome Safari */ -moz-transform: rotate(-3deg); /* 为Firefox */ -ms-transform: rotate(-3deg); /* 为IE */ -o-transform: rotate(-3deg); /* 为Opera */ transform: rotate(-3deg); /* 为nothing */</pre>	

既然 CSS3 代码中（暂时）需要写上这么多前缀，那么就需要注意顺序问题，要先写私有的 CSS3 属性，再写标准的 CSS3 属性。如果以后某个属性成为标准，并且被 Firefox、Chrome 等浏览器的最新版普遍兼容，就可以去除这些前缀了。

6.8.2 边框属性

通过 CSS3，用户能够创建圆角边框，向矩形添加阴影，使用图片来绘制边框。圆角实现是众多 CSS3 属性中很受欢迎的一种，它通过 border-radius 属性来实现，是 CSS3 中级别最高的一个属性。radius 就是“半径”的意思，用这个属性既可以很容易地做出圆角效果，也可以做出圆形效果。原理很简单，“正方形的内切圆的半径等于正方形边长的一半”。边框属性的使用格式如下：

```

-webkit-border-radius: 4px;      /* 适用于WebBit的浏览器 */
-moz-border-radius: 4px;        /* 适用于Firefox浏览器 */
border-radius: 4px;             /* 请在Safari 5和IE 9浏览器中执行这个语法 */

border-radius: 2em;              /* 四角都是圆角 */
/*等价于: */
border-radius: 2em 2em 2em 2em;  /* 四角都是圆角,左上角顺时针 */
/*等价于: */
border-top-left-radius: 2em;     /* 左上角 */
border-top-right-radius: 2em;   /* 右上角 */
border-bottom-right-radius: 2em; /* 右下角 */
border-bottom-left-radius: 2em; /* 左下角 */

/* 根据需要可设置每个角的半径,例如设置左上角的边框半径的css代码如下 */
-webkit-border-top-left-radius: 5px;
-moz-border-radius-topleft: 5px;

```

可以通过 box-shadow 属性向边框添加一个或多个阴影,这也是开发时常用的效果。使用示例如下:

```

<!DOCTYPE html>
<meta charset="utf-8" />
<style>
div {
    width: 300px; height: 100px; border: 1px solid #2cb7fe; text-align: center;
    -webkit-box-shadow: 3px 3px 10px #9edeff;
    box-shadow: 3px 3px 10px #9edeff;
}
</style>

<div>box-shadow:3px 3px 10px #9edeff;</div>

```

box-shadow 属性是由逗号分隔的阴影列表,每个阴影由 2~4 个长度值、可选的颜色值及可选的 inset 关键词来规定,省略长度的值是 0。该属性可用值如下。

- none: 无阴影。
- <length>①: 第 1 个长度值,用来设置对象的阴影水平偏移值。可以为负值。
- <length>②: 第 2 个长度值,用来设置对象的阴影垂直偏移值。可以为负值。
- <length>③: 如果提供了第 3 个长度值,则用来设置对象的阴影模糊值。不允许为负值。
- <length>④: 如果提供了第 4 个长度值,则用来设置对象的阴影外延值。可以为负值。
- <color>: 设置对象的阴影的颜色。
- inset: 设置对象的阴影类型为内阴影。该值为空时,对象的阴影类型为外阴影。

此属性还可以设置多组值,中间用逗号隔开。另外,通过 CSS3 的 border-image 属性,还可以使用图片来创建边框。



6.8.3 背景属性

在 CSS3 中提供了多个背景属性，这里只介绍两个比较常用的属性，其他属性可以通过 CSS3 手册获取帮助。在 CSS3 中，通过 `background-image` 或者 `background` 属性可以为一个容器设置多张背景图片，也就是说，可以把不同的背景图片放到一个块元素中。多张背景图片的 URL 之间使用逗号隔开即可。如果有多张背景图片，而其他属性只有一个，那么所有背景图片都应用该属性值。代码如下：

```
.div1{
    background-image:url(images/1.jpg),url(images/2.jpg),url(images/3.jpg),url(images/4.jpg);
    background-repeat:no-repeat,no-repeat,no-repeat,no-repeat;
    background-position:top left,top right,bottom left,bottom right;
}

/*在上面的代码中有这一句 */
background-repeat:no-repeat,no-repeat,no-repeat,no-repeat;
/*可以简化为 */
background-repeat:no-repeat;

/*上面设置背景图片的各个属性时是分开写的，那么我们也可以把背景图片的各个属性写在一起 */


.div1{
    background:url(images/1.jpg) no-repeat top left,
        url(images/2.jpg) no-repeat top right,
        url(images/3.jpg) no-repeat bottom left,
        url(images/4.jpg) no-repeat bottom right
}


```

背景图片大小调整也是 CSS3 提供的一个新特性，它使得开发人员可以随心所欲地控制背景图片的尺寸大小。在 CSS2 中，背景图片的大小在样式中是不可控的，比如要想使得背景图片充满某个区域，要么重新做一张大一点的图，要么只能让它以平铺的方式来填充。在 CSS3 中提供了 `background-size` 属性，使得开发人员既可以直接缩放背景图片来填充这个区域，也可以设置背景图片的尺寸大小，然后以设置好的尺寸去平铺这个区域。`background-size` 属性需要一个或两个值（一个为必填，一个为可选），这些值既可以是像素（px），也可以是百分比（%）或 auto，还可以是特定值 `cover`、`contain`。示例代码如下：

```
body {
    background: url(path/to/image.jpg) no-repeat;
    -moz-background-size: 100% 100%;
    -o-background-size: 100% 100%;
    -webkit-background-size: 100% 100%;
    background-size: 100% 100%;
}
```

其中，`background-size` 第一个值用于指定背景图片的宽度，第二个值用于指定背景图片的高度。如果只为 `background-size` 设置一个值，则第二个值默认为 auto（`cover` 和 `contain` 除外）。`background-size` 属性的特定值如下。

- **cover**: 保持图片本身的宽高比例, 将图片缩放到完全覆盖定义背景的区域。
- **contain**: 保持图片本身的宽高比例, 将图片缩放到宽度或高度适应定义背景的区域。

6.8.4 文本属性

关于 CSS3 的文本新属性有很多, 在这些属性中常用的有两个: 一个是可以为文字添加阴影的 **text-shadow** 属性, 另一个则是可以强制对单词进行换行处理的 **word-wrap** 属性。**text-shadow** 属性要求的浏览器版本较高, 对于 IE 来说, 至少需要 IE 10 以上版本的支持, 至于 Firefox、Chrome、Safari 及 Opera 等浏览器, 则完全支持这一新属性。大家或许常常借鉴一些国外的主题, 发现有些标题会带有很好看的阴影, 但在 IE 下无法正常地显示出来, 这正是 **text-shadow** 属性在起作用。**word-wrap** 属性算是一个能够被广泛支持的新属性, 几乎所有的主流浏览器都支持这一属性, IE 也不例外。

1. 给文字加上阴影的 text-shadow 属性

text-shadow 属性是我们可以省略前缀的几个属性之一, 这个属性与边框的阴影属性 (**box-shadow**) 类似, 共包含 4 个参数: 水平阴影、垂直阴影、模糊距离及阴影的颜色, 其中前 3 个参数均可以用负值来表示。下面给出了一个简单的代码示例:

```
h1 {  
    text-shadow: 0 1px 0 white;      /* 文本阴影样式 */  
    color: #292929;  
}
```

2. 强制单词换行的 word-wrap 属性

当段落中出现特别长的单词时, 如果没有强制换行, 则可能导致某个单词大量溢出或者直接自动换行而导致行尾留出很大的空白, 这些都使得我们的文本变得很不整齐, **word-wrap** 属性可以用来解决这一问题。请看下面的简单用法:

```
word-wrap: break-word;
```

6.8.5 用户界面属性

在 CSS3 中, 新的用户界面特性包括重设元素尺寸、盒尺寸及轮廓等。本小节着重介绍一下 **resize** 属性, 只有 Firefox 4 和 Safari 3 浏览器支持此属性。**resize** 属性可用于重定义 **textarea** 的大小, 可能的值包括如下几种。

- **none**: UserAgent 没有提供尺寸调整机制, 用户不能调节元素的尺寸。
- **both**: UserAgent 提供了双向尺寸调整机制, 用户可以调节元素的宽度和高度。
- **horizontal**: UserAgent 提供了单向水平尺寸调整机制, 用户可以调节元素的宽度。



➤ **vertical:** UserAgent 提供了单向垂直尺寸调整机制，用户可以调节元素的高度。

示例代码如下：

```
<!-- 默认情况下WebKit浏览器和Firefox 4支持水平和垂直方向上的重定义 -->
<style>
  textarea {
    -moz-resize: vertical;
    -webkit-resize: vertical;
    resize: vertical;
  }
</style>
<textarea name="elem" id="elem" rows="5" cols="50"></textarea>
```

6.8.6 动画属性

也许 CSS3 最令人兴奋的增补，就是在没有 JavaScript 元素的情况下产生动画。CSS3 的动画有 3 个常用属性：transform、transition、animation。transform 属性虽然看起来可以实现动画的效果，但其本质是静态的，其实就是一个图形的变形工具；而 transition 属性是一个简单的动画属性，操作起来非常简单；animation 属性是一个名副其实的动画属性，是 transition 属性的扩展，功能十分强大，可以定义多个关键帧及每个关键帧中元素的属性值来实现复杂的动画效果。下面的示例使用 transition 属性模仿一个效果，当鼠标滑过右侧链接时，文本向右滑动。

```
<style>
ul a {
  -webkit-transition: padding .4s;
  -moz-transition: padding .4s;
  -o-transition: padding .4s;
  transition: padding .4s;
}

a:hover { padding-left: 20px; }
</style>
<ul>
  <li> <a href="#"> Hover Over Me </a> </li>
  <li> <a href="#"> Hover Over Me </a> </li>
  <li> <a href="#"> Hover Over Me </a> </li>
</ul>
```

6.8.7 多列布局属性

通过 CSS3，开发人员能够创建多列来对文本进行布局。在 CSS2 时代，对于多列布局的设计，大多采用浮动布局和绝对定位布局两种方式。浮动布局比较灵活，但是需要编写大

量的附加样式代码，而且在网页缩放等操作下容易发生错位，影响网页整体效果。绝对定位布局方式要精确到标签的位置，但固定标签位置的方式无法满足标签的自适应能力，也影响文档流的联动。CSS3 新增了 columns 属性，即多列自动布局功能，利用该功能可以自动将内容按指定的列数排列（例如，columns: 250px 3, 250px 表示列宽，3 表示多列的数目）。可以结合 column-gap 属性定义列之间的间距，结合 column-rule 属性定义每列之间边框的宽度、样式和颜色。应用示例代码如下：

```
body {
    -webkit-column-count: 3;           /* 定义列数为3列 */
    -webkit-column-gap: 3em;          /* 定义列之间的间距为3em */
    columns: 250px 3;                 /* 定义每列宽度为250px，共定义3列 */
    column-gap: 3em;                  /* 定义列之间的间距为3em */
    -webkit-column-rule: dashed 2px gray; /* 定义列的宽度2px、样式dashed和颜色gray */
    column-rule: dashed 2px gray;      /* 定义列的宽度2px、样式dashed和颜色gray */
}
```

CSS3 多列布局还有很多新属性和特性，如栏目高度（column-fill）属性及分列打印等，这里就不再详述了。

6.8.8 渐变属性

对于正常的渐变背景，一般都是切丝，然后平铺，至于是横向平铺还是纵向平铺，要视实际情况而定，复杂一些的背景就只能切块图。然而 CSS3 可以让 Firefox、Safari、Chrome 实现颜色渐变，IE 可以用滤镜，这也是一种值得考虑的方法。语法如下：

```
.box_gradient {
    background-image: -moz-linear-gradient(top, #444444, #999999);
    background-image: -webkit-gradient(linear, left top, left bottom, color-stop(0, #444444), color-stop(1, #999999));
    /* IE6, IE7 */
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#444444', endColorstr='#999999', GradientType=0);
    /* IE8 */
    -ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr='#444444', endColorstr='#999999', GradientType=0)";
}
```

属性“-moz-linear-gradient”有 3 个参数。第 1 个参数表示线性渐变的方向，top 表示从上到下，left 表示从左到右，如果定义成 left top，则表示从左上角到右下角。第 2 个和第 3 个参数分别表示起点颜色和终点颜色。还可以在它们之间插入更多的参数，表示多种颜色的渐变。

属性“-webkit-gradient”是 WebKit 引擎对渐变的实现，共有 5 个参数。第 1 个参数表示渐变类型（type），可以是 linear（线性渐变）或者 radial（辐射渐变）。第 2 个和第 3 个参数是一对值，分别表示渐变起点和渐变终点。这对值可以用坐标形式表示，也可以用关键值表示，比如 left top（左上角）和 left bottom（左下角）。第 4 个和第 5 个参数分别是两个 color-stop() 函数。color-stop() 函数接受两个参数，第 1 个参数表示渐变的位置，0 为起点，0.5 为中点，1 为结束点；第 2 个参数表示该点的颜色。



IE 依靠滤镜实现渐变。startColorstr 表示起点的颜色，endColorstr 表示终点的颜色。GradientType 表示渐变类型，0 为默认值，表示垂直渐变；1 表示水平渐变。

6.8.9 透明属性

元素透明也是我们常用的样式，在 CSS2 中使用滤镜属性 opacity 实现透明效果。现在有了 CSS3 的 rgba 属性，就不用这么麻烦了，当然也要浏览器支持才行。通常我们定义颜色都用十六进制表示。例如，background:#000000，表示背景色为黑色。当然，也可以用 RGB 三原色值表示，例如，background:rgb(0,0,0)，也表示背景色为黑色。RGBA 只是在 RGB 的基础上增加了一个 A，这个 A 表示透明度。而且这个属性不会被子元素继承下去，可以在定义颜色的属性时使用。应用代码如下：

```
.test{ background:rgba(0,0,0,0.5) }      /* 表示背景色为黑色且半透明 */
.test{ color:rgba(0,0,0,0.5) }          /* 表示字体颜色为黑色且半透明 */
.test{ background:#000000; filter:alpha(opacity=40) } /* IE 滤镜 */
```

除了 IE，其他浏览器几乎都支持 rgba()函数。该函数有 4 个参数，前 3 个参数为一种颜色的 RGB 值，第 4 个参数为透明度。

6.8.10 旋转属性

在 CSS3 中，可以使用 transform 属性对元素进行旋转、缩放、倾斜、平移。以旋转为例，代码如下：

```
div.box_rotate {
    transform:rotate(7.5deg);
    -ms-transform:rotate(7.5deg);      /* IE 9 */
    -moz-transform:rotate(7.5deg);     /* Firefox */
    -webkit-transform:rotate(7.5deg);  /* Safari 和 Chrome */
    -o-transform:rotate(7.5deg);       /* Opera */
}
```

除了早期版本的 IE，其他浏览器都可以使用 rotate()函数来实现某个对象的旋转。比如，rotate(7.5deg)表示顺时针旋转 7.5° (degree)。利用 CSS3，还可以完成 skew（扭曲）和 scale（缩放），以及 css transitions（动态变换）等操作。

6.8.11 服务器端字体属性

设计网页的时候，可能会用到某种特殊的字体。如果用户的计算机中没有安装这种字体，

那么文字只能以普通字体显示。CSS3 可以让用户的浏览器自行下载服务器端字体，从而呈现出设计者想要的效果。应用示例代码如下：

```
@font-face {
    /* 表示为这种字体起一个名称，可以随意设置，我这里用的是MyFont */
    font-family: 'MyFont';
    /* IE6+ 这一行表示字体位置，由于IE只支持服务器端的eot字体，所以这一行是IE专用的 */
    src: url('myfont.eot');
    /* local()表示在本机（客户端）查找该字体，如果本机已经安装了，就不用下载了。
    url()表示字体在服务器上的位置，format()用来说明字体格式。
    Firefox 3.5支持TrueType和OpenType字体，Firefox 3.6又增加了WOFF字体。
    其他基于WebKit引擎的浏览器（Safari、Opera、Chrome），目前好像只支持TrueType */
    src: local('myfont.ttf'),
        url('myfont.woff') format('woff'), /*Firefox 3.6 */
        url('myfont.ttf') format('truetype'); /*Firefox 3.5+, Safari 3+, Chrome, Opera 10+ */
}
/* 使用的时候这样写就可以了 */
h2{ font-family: "MyFont"; }
```

需要注意的是，字体文件必须与网页文件来自同一个域名，符合浏览器的“同源政策”。另外，由于中文字体文件太大，服务器端字体显然只适用于英文字体。

本章小结

CSS3 对于开发者来说，给 web 应用带来了更多的可能性，极大地提高了开发效率。CSS3 在选择器上的支持可谓是丰富多彩，使得我们能够灵活地控制样式，而不必为元素进行规范化的命名。CSS3 支持的动画类型更加丰富，主要有 transform（变换）、transition（过渡）和 animation（动画）。不仅在类型上更加多样，还可以对特定的属性设置 transition，所以在以后面对更多的脚本开发者时，会更加方便、简单。对于 CSS3 的特性，圆角、文本阴影、盒模型阴影和渐变使网页制作更加便捷、网页展示更加丰富。

本章习题

1. 支持 input 类型的输入框的消息提示的属性是（ ）。
 - A. detail
 - B. placeholder
 - C. pattern
 - D. required
2. 以下哪项不属于 HTML5 中<input>标签新增的输入类型？（ ）
 - A. email
 - B. url



- C. number D. radio
3. 对 CSS 编码规范描述不合理的是 ()。
- A. 一般不允许将样式定义写在标签中
B. ID 必须是唯一的, 且用在结构的定义中
C. 尽量不缩写, 除非是一看就明白的单词
D. 建议单位使用绝对长度单位, 如 px、pt 等
4. 在 CSS 中, text-align 属性的初始值是 ()。
- A. start B. 无
C. normal D. auto
5. 下面说法错误的是 ()。
- A. CSS 样式表可以将格式和结构分离
B. CSS 样式表可以控制页面的布局
C. CSS 样式表可以使许多网页同时更新
D. CSS 样式表不能制作体积更小、下载更快的网页
6. 若要设计网页的背景图形为 bg.jpg, 以下标签中正确的是 ()。
- A. <body background="bg.jpg"> B. <body bground="bg.jpg">
C. <body image="bg.jpg"> D. <body bgcolor="bg.jpg">
7. 若要在网页中插入样式表 main.css, 以下用法中正确的是 ()。
- A.<link href="main.css" type=text/css rel=stylesheet>
B.<link Src="main.css" type=text/css rel=stylesheet>
C.<link href="main.css" type=text/css>
D.<include href="main.css" type=text/css rel=stylesheet>
8. 下面不属于 CSS 插入形式的是 ()。
- A. 索引式 B. 内联式
C. 嵌入式 D. 外部式
9. 若要以加粗宋体、10 号字显示“兄弟连 IT 教育”, 以下用法中正确的是 ()。
- A. 兄弟连 IT 教育
B. 兄弟连 IT 教育
C. 兄弟连 IT 教育
D. 兄弟连 IT 教育
10. 若要在当前网页中定义一个独立类的样式 myText, 使具有该类样式的正文字体为“Arial”, 字体大小为 9px, 行间距为 13.5px, 以下定义方法中正确的是 ()。
- A. <style>
 .myText{font-familiy:Arial;font-size:9px;line-height:13.5px}

</style>

B. `.myText{ font-familiy:Arial; font -size:9px;line-height:13.5px}`

C. `<style>`

`.myText{FontName:Arial; fontSize:9px;lineHeight:13.5px}`

`</style>`

D. `<style>`

`.myText{FontName:Arial; font-ize:9px;line-height:13.5px}`

`</style>`

11. 简述题: px 和 em 的区别有哪些?



本章习题及其答案



本章资源包



本章扩展知识

第7章

CSS3 属性特效



对于设计人员和开发人员来说，CSS 一直是 Web 设计过程中重要的一部分。网页外观主要由 CSS 控制，编写 CSS 代码可以任意改变我们的网页布局以及网页内容的样式。随着 CSS3 的出现以及越来越多的浏览器对它的支持，设计师们有了更多的选择。CSS3 可以让网页增添很多动画元素，让我们的网页变得更加生动有趣，并且易于交互。本章利用 CSS3 的样式属性来制作丰富多彩的网页。在本章中，我们对 CSS3 的样式属性作一个介绍，列出该样式属性的属性及用法，使用该样式属性制作一些小案例来对网页样式进行设置，让读者可以更加直观清晰地了解到 CSS3 的样式属性，并能够立即使用它。通过本章的学习，读者可以利用 CSS3 为自己的网站锦上添花。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

7.1 新增颜色模式

CSS3 样式新增了一种颜色模式——RGBA，比 CSS 的颜色模式多了一项透明度的设置。RGB 对于大家来说一点也不陌生，即红色（R）+绿色（G）+蓝色（B）。那现在我们所说的 RGBA 又是什么呢？说得简单一点，就是在 RGB 的基础上加进了一个通道 Alpha。新增颜色模式的参数说明如表 7.1 所示。

表 7.1 CSS3 新增颜色模式的参数说明

属 性	描 述	属性值范围
R	Red (红)	0~255 (0%~100%)
G	Green (绿)	0~255 (0%~100%)
B	Blue (蓝)	0~255 (0%~100%)
A	Alpha (透明度)	0~1

如果说 RGBA 是制作透明色（透明背景色、透明边框色、透明前景色等），那么大家不由会想起 opacity。我们在 CSS2 中制作背景色时通常会用到它，但用它来制作边框色或前景色的话，则无法实现这个功能。这里利用一个实例对 RGB 和 opacity 同时使用与 RGBA 使用作对比，代码如下：

```

1  <html>
2  <head>
3      <meta charset="UTF-8">
4      <title>CSS3 rgba</title>
5      <style>
6          ul{ display:block; width:300px; height:50px; }
7          li{ list-style:none; }
8          .opacity li { float: left; width: 50px; height: 50px; text-align:center;}
9          /* 为该列表添加opacity属性 */
10         .opacity li:nth-of-type(1) { background: rgb(255,255,0); opacity:1; }
11         .opacity li:nth-of-type(2) { background: rgb(255,255,0); opacity:0.8; }
12         .opacity li:nth-of-type(3) { background: rgb(255,255,0); opacity:0.6; }
13         .opacity li:nth-of-type(4) { background: rgb(255,255,0); opacity:0.4; }
14         .opacity li:nth-of-type(5) { background: rgb(255,255,0); opacity:0.2; }
15         .opacity li:nth-of-type(6) { background: rgb(255,255,0); opacity:0; }
16         .rgba li { float: left; width: 50px; height: 50px; text-align:center;}
17         /* 为该列表添加RGBA属性 */
18         .rgba li:nth-of-type(1) { background: rgba(255,255,0,1); }
19         .rgba li:nth-of-type(2) { background: rgba(255,255,0,0.8); }
20         .rgba li:nth-of-type(3) { background: rgba(255,255,0,0.6); }
21         .rgba li:nth-of-type(4) { background: rgba(255,255,0,0.4); }
22         .rgba li:nth-of-type(5) { background: rgba(255,255,0,0.2); }
23         .rgba li:nth-of-type(6) { background: rgba(255,255,0,0); }
24     </style>
25 </head>
26 <body>
27     <div class="example-opacity">
28         <p>CSS3的opacity效果</p>
29         <ul class="opacity">
30             <li>100%</li>
31             <li>80%</li>
32             <li>60%</li>
33             <li>40%</li>
34             <li>20%</li>
35             <li>0%</li>
36         </ul>
37         <p>CSS3的RGBA效果</p>
38         <ul class="rgba">
39             <li>1</li>
40             <li>0.8</li>
41             <li>0.6</li>
42             <li>0.4</li>
43             <li>0.2</li>

```



```
44 <li>0</li>
45 </ul>
46 </div>
47 </body>
48 </html>
```



使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的效果，如图 7.1 所示。我们可以看出，两者的相同之处是背景色完全一样，但区别就是一直让大家觉得头疼的问题，即 `opacity` 后代元素会随着透明值下降一起具有透明性，所以 `opacity` 中的字随着透明值下降越来越看不清楚，而 `RGBA` 不存在这样的问题。

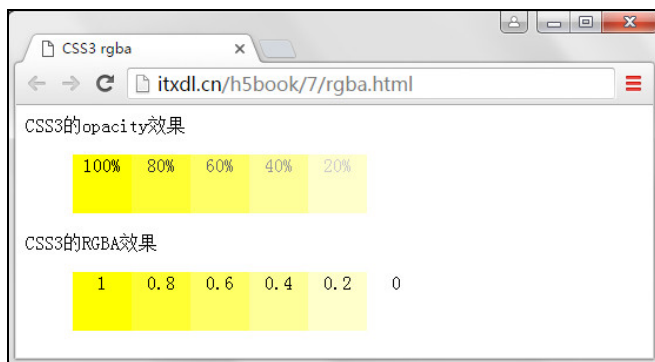


图 7.1 `opacity` 与 `RGBA` 透明度对比

从上面的实例中可以看出，`RGBA` 比为元素设置 `CSS` 的透明度更好。因为单独的颜色不影响整个元素的透明度，也不会影响到元素的其他属性，如边框、字体。同时为某元素设置 `RGBA`，也不会影响到其他元素的相关透明度。

7.2 文字

在 `CSS3` 中，文字也增加了很多特效，为网页增添色彩。文字的 `CSS` 特效有文字阴影、文字描边、文字排版和文字省略等，在下述的内容中笔者会通过小实例一一介绍。下面为读者展示几个比较酷炫的页面，都是关于文字的一些小特效，如图 7.2 所示。

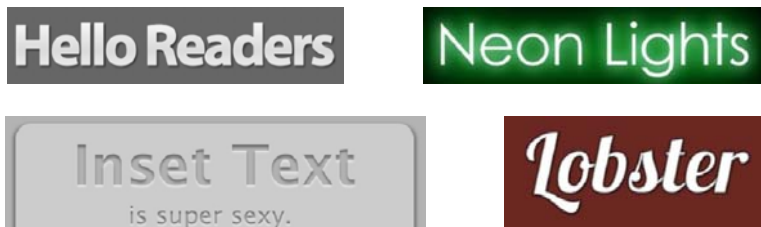


图 7.2 文字的一些特效图

7.2.1 文字阴影

文字阴影是可以叠加的，最基本可以给出四个值，用法如下：

```
text-shadow: x y blur color
```

文字阴影的参数说明如表 7.2 所示。

表 7.2 CSS3 文字阴影的参数说明

属 性	描 述
x	横向偏移
y	纵向偏移
blur	模糊距离
color	阴影颜色

横向偏移量和纵向偏移量可以为负值，代表文字阴影向左偏移或向上偏移。文字阴影是可以叠加的，但是加很多层会影响页面加载速度。添加多层阴影用“,”隔开。阴影叠加是先渲染前面的，再渲染后面的。

1. 最简单的用法

```
text-shadow: 2px 2px 4px #000;
```

此语法说明为一段文字设定横向偏移量为 2px，纵向偏移量为-2px，模糊距离为 3px 的黑色阴影。以下是一个单层阴影的例子。代码如下：

```
1 <html>
2 <head>
3     <meta charset="UTF-8">
4     <title>CSS3单层阴影</title>
5     <style>
6     p{
7         color: #000;
8         background: #fff;
9         /* 为段落添加横向偏移2px, 纵向偏移-2px, 模糊度为3px的黑色阴影 */
10        text-shadow: 2px -2px 3px #000;
11        font-size: 40px;
12    }
13    </style>
14 </head>
15 <body>
16     <p>兄弟连 (itxdl.cn) : </p>
17     <p>三板上市公司亿元级IT教育企业</p>
18 </body>
19 </html>
```





使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的效果，如图 7.3 所示。



图 7.3 文字单层阴影

2. 阴影叠加

```
text-shadow:2px 2px 0px red, 2px 2px 4px green;
```

此语法说明为一段文字设定双层阴影。第一层为横向偏移量为 2px，纵向偏移量为 2px，模糊距离为 0 的红色阴影；第二层为横向偏移量为 2px，纵向偏移量为 2px，模糊距离为 4px 的绿色阴影。对于多层阴影，浏览器先渲染前面的阴影，再渲染后面的阴影。以下是一个双层阴影的例子。代码如下：

```
1 <html>
2 <head>
3   <meta charset="UTF-8">
4   <title>CSS3双层阴影</title>
5   <style>
6     p{
7       color: #000;background: #fff; text-align: center; font-size:40px;
8       /*
9        为段落添加双层阴影，第一层为横向偏移2px，纵向偏移2px模糊度为0的红色阴影，
10      第二层为横向偏移2px，纵向偏移2px模糊度为4px的绿色阴影
11      */
12       text-shadow:2px 2px 0px #e91e63, 2px 2px 4px #673ab7;
13     }
14   </style>
15 </head>
16 <body>
17   <p>兄弟连 (itxdl.cn) </p>
18   <p>股票代码: 839467</p>
19 </body>
20 </html>
```

使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的效果，段落文字被加上了两层阴影，如图 7.4 所示。



图 7.4 文字双层阴影

3. 几个有趣的例子

(1) 层叠:

```

5  <style>
6  p{
7      /* 文字层叠效果 */
8      color: #6479FB; text-align: center; font-size:40px;
9      text-shadow:2px 2px 0px white, 4px 4px 0px #6479FB;
10 }
11 </style>

```

将这段代码写在段落样式表中后，在浏览器中运行，我们就可以看到蓝色层叠的文字效果，如图 7.5 所示。



图 7.5 文字层叠效果

(2) 光晕:

```

5  <style>
6  p{
7      /* 文字光晕效果 */
8      color:white; font-size:40px; line-height:100px; text-align:center;
9      text-shadow:0 0 10px #fff, 0 0 20px #fff, 0 0 30px #fff,
10                 0 0 40px #6479FB, 0 0 70px #6479FB,
11                 0 0 80px #6479FB, 0 0 100px #6479FB,
12                 0 0 150px #6479FB;
13 }
14 </style>

```

将这段代码写在段落样式表中后，在浏览器中运行，我们就可以看到蓝色光晕的文字效果，如图 7.6 所示。



图 7.6 文字光晕效果

(3) 火焰文字:

```
5 <style>
6 body{ background:#FFFC6; }
7 p{
8     /* 火焰文字效果 */
9     font-size:80px; line-height:300px; text-align:center;
10    text-shadow: 0 0 20px #f6cc9, 10px -10px 30px #fec85,
11                -20px -20px 40px #ffae34, 20px -40px 50px #ec760c,
12                -20px -60px 60px #cd4606, 0 -80px 70px #973716,
13                10px -90px 80px #451b0e; color:#fff;
14 }
15 </style>
```



将这段代码写在段落样式表中后，在浏览器中运行，我们就可以看到火焰文字效果，如图 7.7 所示。



图 7.7 火焰文字效果

7.2.2 文字描边

在 CSS3 中，文字描边效果更容易实现，字体及字体颜色可以随意更改。但 IE 9 以下浏览器无效果，所以提醒大家测试时要使用 Google Chrome。-webkit-text-stroke 可以为文字添

加边框。它不但可以设置文字边框的宽度，而且也能设置其颜色。而且，配合使用 `color: transparent` 属性，还可以创建镂空的字体。语法如下：

```
-webkit-text-stroke: 宽度 颜色
```

接下来，为读者准备了一个实例，为段落元素添加文字描边。若浏览器不支持描边，则将段落文字设置为蓝色；若支持，则将段落文字设置为填充颜色为蓝色，描边颜色为黑色。代码如下：

```
1 <html>
2 <head>
3   <meta charset="UTF-8">
4   <title>CSS3文字描边</title>
5   <style>
6     body{ background:#fff; }
7     p{
8       /* 文字描边 */
9       font-family: '微软雅黑';
10      font-size: 3em;
11      color: #6479FB;                               /* 设定文字颜色为蓝色 */
12      text-align: center;
13      -webkit-text-fill-color: #6479FB;               /* 文字内填充颜色为蓝色 */
14      -webkit-text-stroke: 1px black;                 /* 文字内描边颜色为黑色 */
15    }
16  </style>
17 </head>
18 <body>
19   <p>兄弟连IT教育</p>
20   <p>变态严管，让学习成为习惯</p>
21 </body>
22 </html>
```



将这段代码分别在 Chrome、IE 和 Firefox 浏览器中运行，从而可以看到不同的文字效果，只有在 Chrome 浏览器中可以看到文字描边效果，如图 7.8～图 7.10 所示。

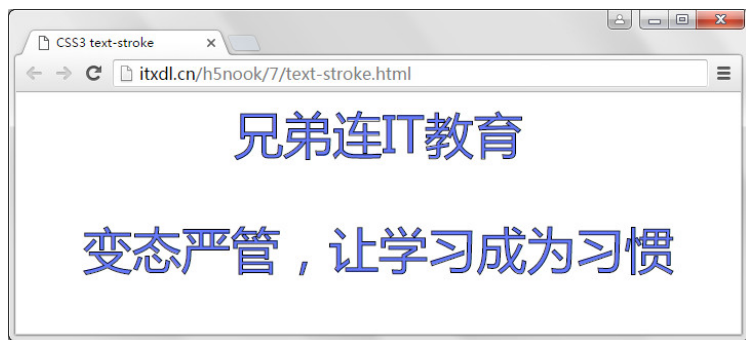


图 7.8 文字描边（Chrome 浏览器）

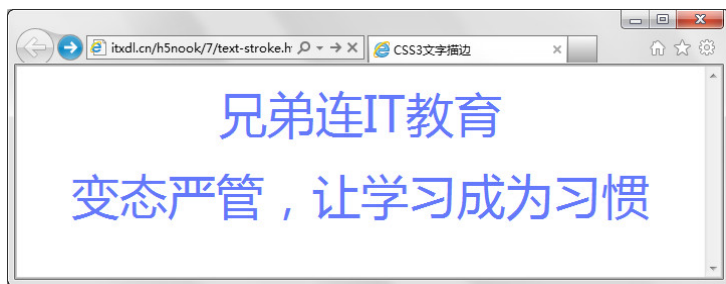


图 7.9 未实现文字描边（IE 浏览器）

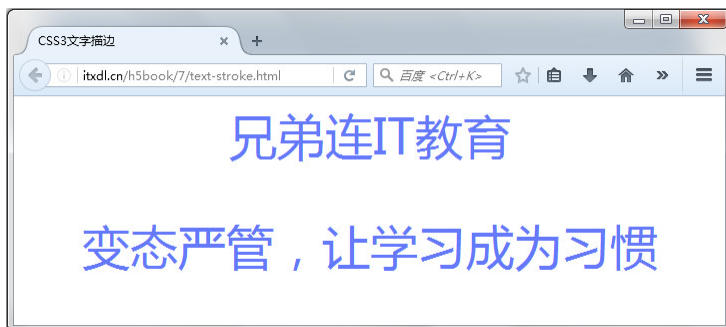


图 7.10 未实现文字描边（Firefox 浏览器）

我们还可以通过文字阴影 `text-shadow` 实现文字描边效果，以下是对使用文字阴影实现文字描边效果的几条解释。

- (1) `text-shadow`: 向文本设置阴影。
- (2) `text-shadow: h-shadow v-shadow blur color`。
- (3) `h-shadow`: 指定阴影在水平方向上的延伸距离，可以为负值。
- (4) `v-shadow`: 指定阴影在垂直方向上的延伸距离，可以为负值。
- (5) `blur`: 指定阴影模糊效果的作用距离。
- (6) 用空格分隔的 4 个属性值代表的方向顺序为右下左上。
- (7) 为了兼容多浏览器而加上前缀 `-webkit-` 和 `-moz-`。

现在想要使用 `text-shadow` 实现文字描边效果，如上例中那样，同样设置文字颜色为蓝色，描边颜色为黑色，代码如下：

```
<style>
body{ background:#fff; }
p{
    /* 文字描边 */
    font-family: '微软雅黑';
    font-size: 3em;
    color: #6479FB;
    text-align: center;
```



```

/* 为段落文字设置颜色为蓝色，边框为1px黑色，设置顺序为右下上左 */
text-shadow:2px 0 0 #000,0 2px 0 #000,-2px 0 0 #000,0 -2px 0 #000;
-webkit-text-shadow:2px 0 0 #000,0 2px 0 #000,-2px 0 0 #000,0 -2px 0 #000;
-moz-text-shadow:2px 0 0 #000,0 2px 0 #000,-2px 0 0 #000,0 -2px 0 #000;
}
</style>

```

将这段代码写在段落样式表中后，分别在 Chrome、IE 和 Firefox 浏览器中运行，从而可以看到不同的文字效果，在 Chrome 和 Firefox 浏览器中可以看到文字描边效果，如图 7.11～图 7.13 所示。



图 7.11 利用 text-shadow 实现文字描边效果（Chrome 浏览器）



图 7.12 利用 text-shadow 实现文字描边效果（IE 浏览器）



图 7.13 利用 text-shadow 实现文字描边效果（Firefox 浏览器）



通过上述两个例子的对比，我们可以发现，使用 `-webkit-text-stroke` 和 `text-shadow` 都可以实现文字描边效果。但是 `-webkit-text-stroke` 只支持 webkit 内核的浏览器，而 `text-shadow` 支持 webkit 和 moz 内核的浏览器。而 `-webkit-text-stroke` 设置描边会比较简单。通过仔细对比我们可以发现，两者实现描边的效果有细微的差异。在 `-webkit-text-stroke` 例子中，我们将这个属性的宽度设置为 `2px`，而 `text-shadow` 的延伸距离也设置为 `2px`。但是 `text-shadow` 设置的描边明显宽于 `-webkit-text-stroke`，原因是 `text-shadow` 的描边是向外延伸的。`-webkit-text-stroke` 实现的描边效果更好，使用方法也更加简单，所以建议读者在文字描边时使用 `-webkit-text-stroke`。

`-webkit-text-stroke` 配合使用 `color:transparent` 属性，还可以创建镂空的字体。下面实现一个蓝色的镂空字体，代码如下：

```
1 <html>
2 <head>
3   <meta charset="UTF-8">
4   <title>CSS3文字镂空</title>
5   <style>
6     body{ background:#fff; }
7     h1{
8       font-size:6em;
9       text-align: center;
10      /* 为一级标题设置镂空字体 */
11      color: transparent;
12      -webkit-text-stroke: 4px #6479FB;
13    }
14  </style>
15 </head>
16 <body>
17   <h1>CSS3镂空</h1>
18 </body>
19 </html>
```



在 Chrome 浏览器中运行这段代码，可以看到文字镂空效果，如图 7.14 所示。



图 7.14 文字镂空效果

7.2.3 文字排版

`direction` 定义文字排列方式，所有浏览器都兼容这个属性，有两个可选值 `rtl` 和 `ltr`。文字排版的参数说明如表 7.3 所示。

表 7.3 CSS3 文字排版的参数说明

属 性	描 述
rtl	从右向左排列（注意要配合 unicode-bidi 一起使用）
ltr	从左向右排列（默认）

ltr 是初始值，表示 left-to-right，即从左向右的意思，具体描述就是内联内容从左向右依次排列。我们平时网页的处理都是这样的，比如说前后两张图片，在默认情况下，DOM 在前的就显示在左边。

rtl 则是另外一个值，是 right-to-left 的缩写，即从右向左的意思。再具体一些描述就是，内联内容是从右向左依次排列的，应用这个 CSS 声明后，则前后两张图片，在默认情况下，DOM 在前的就显示在右侧，并且是在容器的右端。

我们可以通过 `direction` 为段落文字进行排版。本例的段落文字默认是从左向右排版，当鼠标移入该元素时，将排版方向变为从右向左排版，当鼠标移开时恢复为从左向右排版。代码如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5     <title>文字排版text-direction</title>
6     <style>
7     {
8         background-color:#6479FB; color:#fff;text-align:center;font-size:25px;
9         /* 设定文字为从左向右排布 */
10        direction:ltr;unicode-bidi:bidirectional;
11    }
12    /* 当鼠标移入段落时，文字从左向右排布 */
13    p:hover{ direction:rtl; }
14 </style>
15 </head>
16 <body>
17 <p>鼠标移入，文字从左向右排布，移除后从右向左排布</p>
18 </body>
19 </html>

```

使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的效果，段落文字默认从左向右排版，如图 7.15 所示。



图 7.15 文字从左向右排版



当鼠标移入段落元素时，段落文字变为从右向左排版，如图 7.16 所示。



图 7.16 文字从右向左排版

7.2.4 定义省略文本的处理方式

`text-overflow` 属性仅是注解，当文本溢出时是否显示省略标记，并不具备其他的样式属性定义。我们想要实现溢出时产生省略号的效果。还必须定义：强制文本在一行内显示（`white-space:nowrap`）及溢出内容为隐藏（`overflow:hidden`）。只有这样，才能实现溢出文本显示省略号的效果。

省略文本的参数说明如表 7.4 所示。

表 7.4 省略文本的参数说明

属 性	描 述
clip	不显示省略标记(...), 而是简单裁切
ellipsis	当对象内文本溢出时显示省略标记(...)

如果想让某个容器（`div` 或 `li` 或块级元素）显示一行文字，当文字内容过多时不换行，而是出现“...”，则应注意以下几点：

- 仅定义 `text-overflow:ellipsis`;不能实现省略号效果。
- 定义 `text-overflow:ellipsis`; `white-space:nowrap`;同样不能实现省略号效果。
- 同时应用 `text-overflow:ellipsis`; `white-space:nowrap`; `overflow:hidden`;可实现所想要得到的溢出文本显示省略号效果。

我们可以通过 `text-overflow` 为列表限制长度，超出长度的文字用省略号代替。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>文字溢出省略</title>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <style>
7 h2{ text-align:center;}
8 ul{
9     border:1px #ff8000 solid; padding:20px; width:500px; margin:0px auto;
```

```

10 }
11 ul li{
12     list-style:none;width:500px;height:24px;line-height:24px;
13     font-size:15px;color:#6699ff;border-bottom:1px #ccc dashed;
14     white-space:nowrap;overflow:hidden;text-overflow:ellipsis;;
15 }
16 </style>
17 </head>
18 <body>
19     <h2>兄弟连IT教育 (itxdl.cn) </h2>
20     <ul>
21         <li>
22             三板上市公司  亿元级IT教育企业
23         </li>
24         <li>
25             靠谱的教学，严格的管理，职业素质课贯穿始终
26         </li>
27         <li>
28             国内专业PHP培训学校：兄弟连于2006年成立，十年专注于PHP培训，
29             是国内专业PHP培训学校，堪称PHP界的黄埔军校。
30         </li>
31     </ul>
32 </body>
33 </html>

```



使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，如图7.17所示。



图 7.17 文字溢出省略效果

7.3 自定义文字

字体使用是网页设计中不可或缺的一部分。经常地，我们希望在网页中使用某一特定字体，但是该字体并非主流操作系统的内置字体，这样用户在浏览页面的时候就有可能看不到真实的设计。美工设计师最常用的办法是把想要的文字做成图片，但这样做有几个明显的缺陷：



- 不能大范围地使用该字体。
- 图片内容相对使用文字不易修改。
- 不利于网站搜索引擎优化（主流搜索引擎不会将图片 alt 内容作为判断网页内容相关性的有效因素）。

网络上有一些使用 sIFR 技术或 JavaScript/flash hack 的方法，但实现起来或烦琐，或有缺陷。下面要讲的是如何只通过 CSS 的 @font-face 属性来实现在网页中嵌入任意字体。

首先获取要使用字体的 3 种文件格式，确保在主流浏览器中都能正常显示该字体。

- TTF 或 OTF，适用于 Firefox 3.5、Safari、Opera。
- EOT，适用于 Internet Explorer 4.0+。
- SVG，适用于 Chrome、iPhone。

下面要解决的是如何获取某种字体的这 3 种格式文件。一般地，我们在手头上（或在设计资源站点已经找到）有该字体的某种格式文件，最常见的是 TTF 文件，我们需要通过对这种文件格式进行转换来得到其余两种文件格式。字体文件格式的转换可以通过网站 FontsQuirrel 或 onlinefontconverter 提供的在线字体转换服务实现。这里推荐第一个站点，它允许我们选择需要的字符生成字体文件（在服务的最后一个选项），从而大大缩减了字体文件的大小，使得本方案更具实用性。字体声明如下。

（1）下载一种字体到本地后引入该字体，为该字体命名：

```
@font-face{
    font-family: 'fontNameRegular';
    src: url('fontName.eot');
    src: local('fontName Regular'),
        local('fontName'),
        url('fontName.woff') format('woff'),
        url('fontName.ttf') format('truetype'),
        url('fontName.svg#fontName') format('svg');
}
```

（2）在页面中需要的地方使用该字体：

```
h1{font-family: fontNameRegular;}
p { font: 13px fontNameRegular, Arial, sans-serif; }
```

下面的例子是对上述步骤的具体实施，也是对自定义文字进行进一步说明，方便读者理解及使用自定义文字。这里我们在网上下载了“hanyi.ttf”字体，因为该字体不是系统字体，所以我们不能直接使用“font-family”方法为网页设置该字体。这时候就需要使用 CSS3 自定义文字的方法来使用该字体，我们将该文字命名为“itxdl”后就可以用“font-family”方法来导入该字体了。代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>自定义文字</title>
6   <style>
7     @font-face{
8       font-family: 'itxdl';
9       src: url('../fonts/hanyi.ttf') format('truetype');
10      font-weight: normal;
11      font-style: normal;
12    }
13    h1{ font-size: 28px; font-family: itxdl}
14    p { font: 22px itxdl, Arial, sans-serif; }
15  </style>
16 </head>
17 <body>
18   <div id="box">
19     <h1>兄弟连IT教育</h1>
20     <p>
21       兄弟连IT教育隶属于易第优（北京）教育咨询股份有限公司，
22       成立于2006年（以下简称兄弟连）。专注于IT技术培训，
23       是国内专业的PHP/LAMP技术专业培训学校。
24     </p>
25   </div>
26 </body>
27 </html>

```



使用 Chrome 浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，如图 7.18 所示。通过 CSS3 自定义文字的设置，我们可以看到网页上字体已经变成“hanyi.ttf” 字体样式了。



图 7.18 使用自定义文字

使用这个方法，我们就可以为自己的页面设置我们想要的个性化文字了。是不是很激动，想要为自己的网站设置早就看好的文字呢？那就赶紧试试吧！



7.4 弹性盒模型

CSS3 引入了新的盒模型——弹性盒模型，该模型决定一个盒子在其他盒子中的分布方式以及如何处理可用的空间。使用该模型，可以很轻松地创建自适应浏览器窗口的流动布局或自适应字体大小的弹性布局。弹性盒模型看起来很不错，Gecko 和 WebKit 对该模型都有一些尝试性的测试。在这些属性之前加上-moz-和-webkit-即可使用该属性。也就是说，Firefox、Safari、Chrome 可以使用这些特性。该模型对我们解决网页设计中一些常见的问题非常有帮助，如表单布局、垂直居中、视觉上分离 HTML 流等。为了方便，笔者在这里都使用 WebKit 的前缀来讲解弹性盒模型的例子，读者可以通过 Chrome 浏览器预览案例。弹性盒模型的例子使用以下的 HTML 代码：

```
7 <body>
8   <div id="box">
9     <div>1</div>
10    <div>2</div>
11    <div>3</div>
12    <div>4</div>
13    <div>5</div>
14  </div>
15 </body>
```

传统的盒模型基于 HTML 流在垂直方向上排列盒子。使用弹性盒模型可以规定特定的顺序，也可以反转之。要开启弹性盒模型，只需设置拥有子盒子的盒子的 display 的属性值为 box（或 inline-box）即可。这里的盒子是 box，为 box 开启盒模型：

```
#box{ display:box; } /* 为 box 盒子开启盒模型 */
```

弹性盒子基础布局的 CSS 代码如下：

```
6 <style>
7 #box{ width:100%; height:100px; padding:2px 0; border:1px solid #000; display:-webkit-box; }
8 #box div{ width:100px; height:100px; margin:0 1px; font-size:20px; }
9 #box div:nth-of-type(1){ background:#FECCCB; }
10 #box div:nth-of-type(2){ background:#CDFFCF; }
11 #box div:nth-of-type(3){ background:#CECCFE; }
12 #box div:nth-of-type(4){ background:#CDFFCF; }
13 #box div:nth-of-type(5){ background:#FECCCB; }
14 </style>
```



使用 Chrome 浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，如图 7.19 所示。

1. box-orient 定义盒模型的布局方向

box-orient 的参数说明如表 7.5 所示。

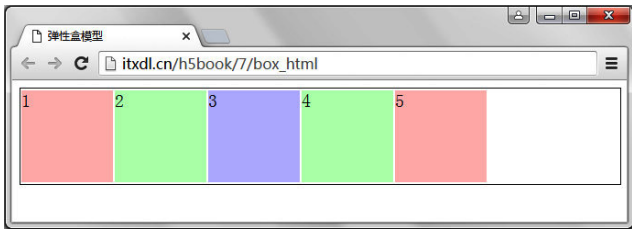


图 7.19 弹性盒模型

表 7.5 box-orient 的参数说明

参 数	描 述
horizontal	水平显示
vertical	垂直显示

box-orient 定义分布的坐标轴：vertical 和 horizontal。默认是 horizontal 水平，当将 box 中的该属性设为水平方向 vertical 时，box 的子盒子垂直显示，为 box 设置 CSS 部分代码如下：

```
7  #box{
8      width:100%; height:100px; padding:2px 0; border:1px solid #000;
9      display:-webkit-box; -webkit-box-orient:vertical;
10 }
```



使用 Chrome 浏览器重新打开这个文件，就可以看到这个盒模型的子盒子垂直分布，如图 7.20 所示。

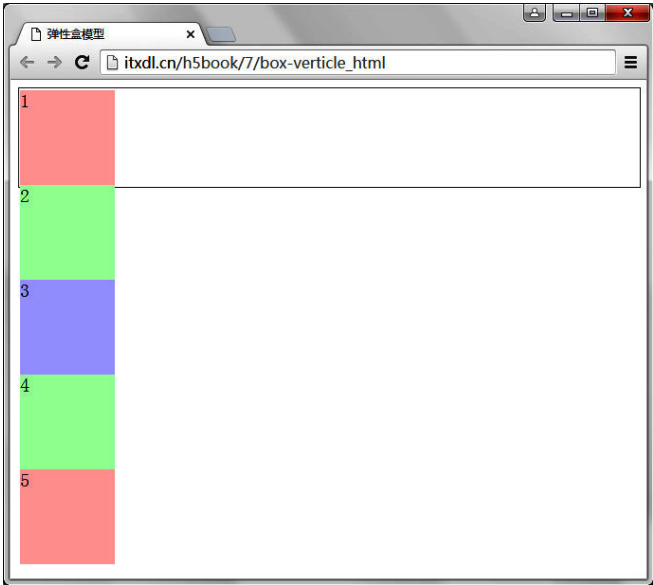


图 7.20 弹性盒模型垂直分布



2. box-direction 元素排列顺序

box-direction 可以设置盒子出现的顺序。默认情况下，只需定义分布坐标轴——box 随 HTML 流分布。如果为水平坐标轴，则从左到右分布；垂直坐标轴则从上到下分布。box-direction 的参数说明如表 7.6 所示。

表 7.6 box-direction 的参数说明

参 数	描 述
normal	正序
reverse	反序

定义 box-direction 的属性值为“reverse”，则反转盒子的排列顺序。若不设置该属性值或将该属性值设置为“normal”，则盒子以正序排列。接下来，我们将盒子设置为水平分布并将其出现的顺序反转，box 的 CSS 代码如下：



```
7 #box{
8   width:100%; height:100px; padding:2px 0; border:1px solid #000;
9   display:-webkit-box; -webkit-box-orient:horizontal; -webkit-box-direction: reverse;
10 }
```

将上例的 box 的 CSS 代码替换后，使用 Chrome 浏览器重新打开这个文件，就可以看到这个盒模型的顺序反转了，如图 7.21 所示。

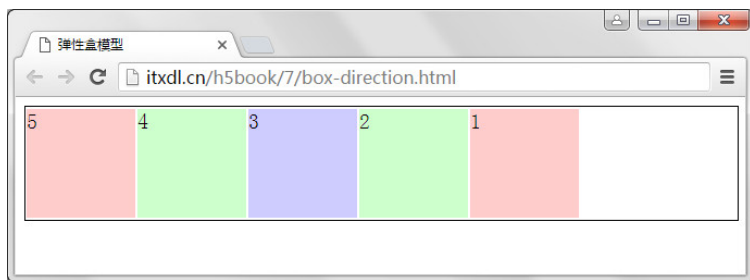


图 7.21 弹性盒模型反转

3. box-ordinal-group 设置元素的具体位置

属性 box-ordinal-group 定义盒子分布的顺序。可以随意控制其分布顺序。这些组以一个从“1”开始的数字定义，盒模型将首先分布这些组，所有这些盒子将在每个组中。分布将从小到大排列。将盒子的分布顺序按照特定的顺序排列，如“3—2—1—5—4”，CSS 代码如下：

```
6 <style>
7 #box{
8   width:100%; height:100px; padding:2px 0; border:1px solid #000;
9   display:-webkit-box;
10 }
11 #box div{ width:100px; height:100px; margin:0 1px; font-size:20px; }
12 #box div:nth-of-type(1){ -webkit-box-ordinal-group: 3; background:#FECCCB; }
```



```

13 #box div:nth-of-type(2){ -webkit-box-ordinal-group: 2; background:#CDFFCC; }
14 #box div:nth-of-type(3){ -webkit-box-ordinal-group: 1; background:#CECCFE; }
15 #box div:nth-of-type(4){ -webkit-box-ordinal-group: 5; background:#CDFFCC; }
16 #box div:nth-of-type(5){ -webkit-box-ordinal-group: 4; background:#FECCCB; }
17 </style>

```

将上例的 CSS 代码替换，使用 Chrome 浏览器重新打开这个文件，就可以看到这个盒模型已经按照“3—2—1—5—4”的顺序排列，如图 7.22 所示。

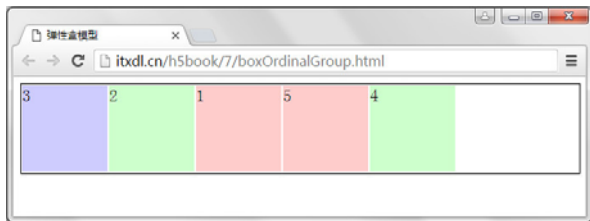


图 7.22 弹性盒模型排序

4. box-flex 定义盒子的弹性空间

默认情况下，盒子并不具有弹性，如果 box-flex 的属性值至少为 1，则盒子变得富有弹性。如果盒子是弹性的，则其大小将按下面的公式计算：

子元素的尺寸=盒子的尺寸×子元素的 box-flex 属性值/所有子元素的 box-flex 属性值的和

在下面的例子中，为每个盒子设定弹性空间。看起来好像是用百分比定义盒子的大小，但区别是，使用弹性盒模型，增加一个盒子，无须重新计算其大小。CSS 代码如下：

```

6 <style>
7 #box{
8     width:100%; height:100px; padding:2px 0; border:1px solid #000;
9     display:-webkit-box;
10 }
11 #box div{ width:100px; height:100px; margin:0 1px; font-size:20px; }
12 #box div:nth-of-type(1){ -webkit-box-flex:1; background:#FECCCB; }
13 #box div:nth-of-type(2){ -webkit-box-flex:2; background:#CDFFCC; }
14 #box div:nth-of-type(3){ -webkit-box-flex:3; background:#CECCFE; }
15 #box div:nth-of-type(4){ -webkit-box-flex:2; background:#CDFFCC; }
16 #box div:nth-of-type(5){ -webkit-box-flex:1; background:#FECCCB; }
17 </style>

```



将上例的 CSS 代码替换，使用 Chrome 浏览器重新打开这个文件，就可以看到每个盒子的弹性空间不同，如图 7.23 所示。

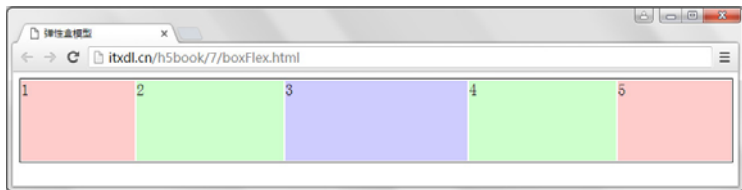


图 7.23 盒模型弹性空间（1）



除了为每个盒子设定弹性空间，我们还可以为部分盒子设定弹性空间，而其他部分盒子有固定大小。例如，我们将中间的盒子，也就是第三个盒子设定一个固定宽度为 200px，其他盒子设定 `box-flex` 的属性值至少为 1 时，随着 `box` 盒子的宽度变化。除了第三个子盒子的宽度固定，其他盒子的宽度都会变化。CSS 代码如下：

```
6 <style>
7 #box{
8     width:100%; height:100px; padding:2px 0; border:1px solid #000;
9     display:-webkit-box;
10 }
11 #box div{ width:100px; height:100px; margin:0 1px; font-size:20px; }
12 #box div:nth-of-type(1){ -webkit-box-flex:1; background:#FECCEB; }
13 #box div:nth-of-type(2){ -webkit-box-flex:2; background:#CDE9F6; }
14 #box div:nth-of-type(3){ width:200px; background:#FECCEB; }
15 #box div:nth-of-type(4){ -webkit-box-flex:2; background:#CDE9F6; }
16 #box div:nth-of-type(5){ -webkit-box-flex:1; background:#FECCEB; }
17 </style>
```



将上例的 CSS 代码替换，使用 Chrome 浏览器重新打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，如图 7.24 所示。

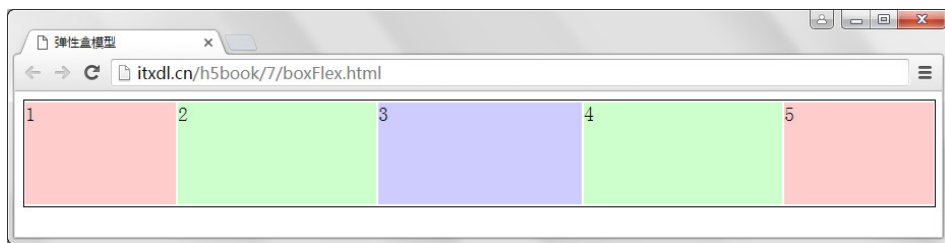


图 7.24 盒模型弹性空间（2）

我们试图改变 `box` 的宽度来查看子盒子的大小，因为这里设定的 `box` 宽度为 100%，所以我们可以通过改变浏览器的大小来控制 `box` 的宽度。当我们缩小浏览器时，可以观察到第三个盒子的宽度固定，其他盒子的宽度变小，如图 7.25 所示。

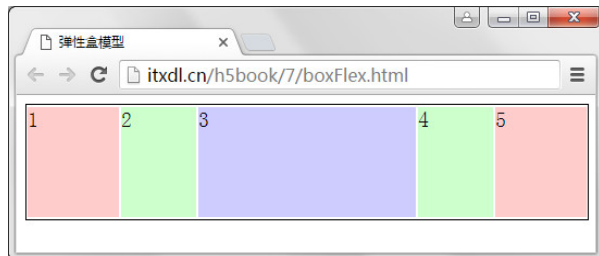


图 7.25 盒模型弹性空间（3）

同样，我们也可以将两边的盒子固定，中间盒子的宽度随浏览器的大小而变化，这里需要读者自己去实践练习。

5. box-pack 对盒模型富裕空间进行管理

box-pack 属性规定当框大于子元素的尺寸时，在何处放置子元素。该属性规定水平框中的水平位置，以及垂直框中的垂直位置。box-pack 的参数说明如表 7.7 所示。

表 7.7 box-pack 的参数说明

参 数	描 述
start	所有子元素在盒子左侧显示，富裕空间在右侧
end	所有子元素在盒子右侧显示，富裕空间在左侧
center	所有子元素居中
baseline	如果 box-orient 是 inline-axis 或 horizontal，则所有子元素均与其基线对齐
justify	富裕空间在子元素之间平均分布

下面通过一个案例来对盒模型的富裕空间进行管理，通过一起使用 box-align 和 box-pack 属性，当有富裕空间时，让所有子元素居中。CSS 代码如下：

```
6 <style>
7 #box{
8     width:100%; height:100px; padding:2px 0; border:1px solid #000;
9     display:-webkit-box; -webkit-box-pack:center;
10 }
11 #box div{ width:100px; height:100px; margin:0 1px; font-size:20px; }
12 #box div:nth-of-type(1){ background:#FECCCB; }
13 #box div:nth-of-type(2){ background:#CDFFCC; }
14 #box div:nth-of-type(3){ background:#CECCFE; }
15 #box div:nth-of-type(4){ background:#CDFFCC; }
16 #box div:nth-of-type(5){ background:#FECCCB; }
17 </style>
```



将上例的 CSS 代码替换，使用 Chrome 浏览器重新打开这个文件，就可以看到盒子的子元素居中，如图 7.26 所示。

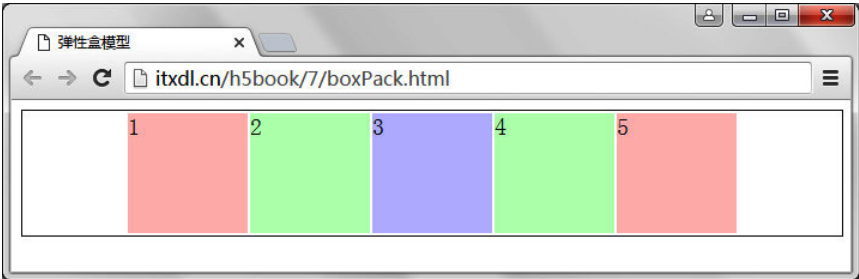


图 7.26 盒模型富裕空间管理

特别地，我们来对 box-align 属性的“baseline”属性值来进行说明。baseline 表示设置伸缩盒对象的子元素基线对齐，可能大多数读者不理解什么叫作“基线对齐”，我们通过下面



的案例来对这个属性值进行图形化的理解，代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>盒模型子元素基线对齐</title>
6   <style>
7     #box{
8       width:400px; height:200px; padding:0 2px; border:1px solid #000;
9       display:-webkit-box;margin:0 auto;
10      /* 设置伸缩盒对象的子元素基线对齐 */
11      -moz-box-align:baseline; -webkit-box-align:baseline;
12      -o-box-align:baseline; box-align:baseline;
13    }
14    #box div{
15      width:100px; height:130px; line-height: 130px;
16      font-size:20px; text-align: center;
17    }
18    #box div:nth-of-type(1){ background:#FECCCB; }
19    #box div:nth-of-type(2){ background:#CDEFFC; line-height: 200px; }
20    #box div:nth-of-type(3){ background:#CECCFE; }
21    #box div:nth-of-type(4){ background:#CDEFFC; line-height: 100px; }
22    #box div:nth-of-type(5){ background:#FECCCB; }
23  </style>
24 </head>
25 <body>
26 <div id="box">
27   <div>1</div>
28   <div>2</div>
29   <div>3</div>
30   <div>4</div>
31   <div>5</div>
32 </div>
33 </body>
34 </html>
```



本例中的盒模型包含 5 个子元素，并将该盒子的 box-align 属性设置为基线对齐“baseline”。将第 2 个子元素的行高设为 200px，第 4 个子元素的行高设为 100px，其余子元素为本身的高度 130px。运行这个文件，得到的效果如图 7.27 所示。



图 7.27 盒模型子元素基线对齐

通过本例的展示，我们不难理解基线对齐的意义，它根据盒模型子元素的主体内容进行对齐。

6. box-align 在垂直方向上对元素的位置进行管理

box-align 属性规定如何对齐框的子元素。box-align 的参数说明如表 7.8 所示。

表 7.8 box-align 的参数说明

参 数	描 述
start	所有子元素在盒子顶部显示，富裕空间在底部
end	所有子元素在盒子底部显示，富裕空间在顶部
center	所有子元素垂直居中

下面通过一个案例来对盒模型的垂直方向的富裕空间进行管理。当垂直方向有富裕空间时，让所有子元素居中，HTML 代码如下：

```
1  <!doctype html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>弹性盒模型</title>
6  <style>
7      #box{
8          width:400px; height:200px; padding:0 2px; border:1px solid #000;
9          display:-webkit-box;margin:0 auto;
10         -webkit-box-align: center;
11     }
12     #box div{ width:100px; height:100px; margin:1px 0; font-size:20px; }
13     #box div:nth-of-type(1){ background:#FECCCB; }
14     #box div:nth-of-type(2){ background:#CDFFCF; }
15     #box div:nth-of-type(3){ background:#CECCFE; }
16 </style>
17 </head>
18 <body>
19     <div id="box">
20         <div>1</div>
21         <div>2</div>
22         <div>3</div>
23     </div>
24 </body>
25 </html>
```



使用 Chrome 浏览器重新打开这个文件，就可以看到盒模型的子元素居中显示，如图 7.28 所示。

作为前端开发者来说，该模型对我们解决网页设计中一些常见的问题非常有帮助，如表单布局、垂直居中、视觉上分离 HTML 流等。



图 7.28 盒模型垂直富裕空间管理

7.5 盒模型阴影

除了为文字添加阴影，我们还可以为盒模型添加阴影。盒模型阴影的属性名称为 `box-shadow`，此属性与 `text-shadow` 一样有 4 个值，前两个值分别表示水平方向位移距离和垂直方向的位移距离，第三个值表示阴影的模糊半径（包含 0 及以下的值均表示无模糊），最后一个值则是阴影的颜色。Chrome 16+、Firefox 8+、Opera 11.6+、Safari 5.1+以及 IE 9+ 均可直接使用 `box-shadow`，而不需要诸如 `-webkit-` 之类的前缀。

语法格式如下：

```
box-shadow:[inset] x y blur [spread] color
```

`box-shadow` 的参数说明如表 7.9 所示。

表 7.9 `box-shadow` 的参数说明

参 数	描 述
inset	投影方式（inset：内投影，不指定：外投影）
x	阴影水平方向偏移量
y	阴影垂直方向偏移量
blur	模糊半径
spread	扩展阴影半径（先扩展原有形状，再开始画阴影）
color	阴影颜色

与文字阴影 `text-shaow` 的参数意义一致，横向偏移量和纵向偏移量可以为负值，代表盒

模型阴影向左偏移或向上偏移。盒模型阴影也可以叠加，添加多层阴影用“,”隔开。阴影叠加是先渲染后面的，再渲染前面的。

1. 最简单的用法

```
box-shadow: 2px 2px 20px #000;
```

此语法说明为一段文字设定横向偏移量为 2px，纵向偏移量为 2px，模糊距离为 20px 的黑色阴影。以下是一个单层阴影的例子。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>box-shadow盒模型阴影</title>
6     <style>
7       #box{
8         width:200px; height:200px; text-align:center; line-height:200px; margin:10px auto;
9         /* 为盒子添加单层黑色阴影，横向和纵向偏移量为2px，模糊距离为20px */
10        box-shadow: 2px 2px 20px #000;
11      }
12    </style>
13  </head>
14  <body>
15    <div id="box">兄弟连IT教育</div>
16  </body>
17 </html>
```



使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，可以看到 box 的外面增加了一层黑色的阴影，如图 7.29 所示。



图 7.29 盒模型阴影

2. 盒模型阴影的投影方式

```
box-shadow: inset 2px 2px 20px #000;
```

此语法说明为盒模型嵌套一层内投影，即横向偏移量为 2px，纵向偏移量为 2px，模糊距离为 20px 的黑色内投影。代码如下：



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>box-shadow盒模型内投影</title>
6     <style>
7       #box{
8         width:200px; height:200px; text-align:center;line-height:200px; margin:10px auto;
9         /* 为盒子添加单层的黑色内投影，横向和纵向的偏移量为2px，模糊距离为20px */
10        box-shadow: inset 2px 2px 20px #000;
11      }
12    </style>
13  </head>
14  <body>
15    <div id="box">兄弟连(itxdl.cn)三板上市</div>
16  </body>
17 </html>
```



使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，盒模型被加上了内投影，如图 7.30 所示。



图 7.30 盒模型内投影

3. 盒模型阴影叠加

```
box-shadow: 2px 2px 20px green, inset 2px 2px 20px blue;
```

此语法说明为盒模型设定双层阴影。第一层为横向偏移量为 2px，纵向偏移量为 2px，模糊距离为 20px 的向外的绿色阴影；第二层为横向偏移量为 2px，纵向偏移量为 2px，模糊距离为 20px 的蓝色内投影。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>box-shadow盒模型阴影叠加</title>
6     <style>
7       #box{
8         width:200px; height:200px; text-align:center;line-height:200px; margin:10px auto;
9         /* 为盒模型设定双层阴影。第一层为横向偏移量和纵向偏移量都为2px，模糊距离为20px的向外的绿色阴影，*/
10        /* 第二层为横向偏移量为2px，纵向偏移量为2px，模糊距离为20px的蓝色内投影。 */
11        box-shadow: 2px 2px 20px green, inset 2px 2px 20px blue;
12      }
13    </style>
14  </head>
15  <body>
16    <div id="box">兄弟连(itxdl.cn)三板上市</div>
17  </body>
18 </html>
```



```

13     </style>
14 </head>
15 <body>
16     <div id="box">盒模型阴影叠加</div>
17 </body>
18 </html>

```

使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，box 被加上了两层阴影，其中有一层是内投影，如图 7.31 所示。



图 7.31 盒模型阴影叠加

7.6 倒影

在 Web 制作中，有些时候需要实现一些倒影的效果。在传统网页中，我们只能使用 Photoshop 事先将倒影设计好，然后再导入网页中，这样不但耗费资源，也阻碍了开发效率。而 CSS 新增了 Reflections 板块，CSS Reflections 允许设计倒影。目前，CSS Reflections 仅获得了 WebKit 引擎的支持，我们只能在 Chrome 和 Safari 浏览器中进行测试。CSS3 的 box-reflect 属性，使我们可以对图片、文字等进行倒影设计，具体语法如下：

```
box-reflect: none | <direction> <offset> ? <mask-image> ?
```

由于此属性并不是 W3C 标准属性，在具体使用之时，还需要添加浏览器的私有属性。根据浏览器的兼容性，使用 box-reflect 时需要添加-webkit-的前缀。

1. direction

direction 定义方向，取值包括 above、below、left、right，具体含义如表 7.10 所示。



表 7.10 box-reflect 属性的 direction 参数值取值说明

属性值	描 述
above	指定倒影在对象的上边
below	指定倒影在对象的下边
left	指定倒影在对象的左边
right	指定倒影在对象的右边

2. offset

offset 定义反射偏移的距离，取值包括数值或百分比，其中百分比根据对象的尺寸进行确定，默认为 0。用长度值来定义倒影与对象之间的间隔，可以为负值；或者用百分比来定义倒影与对象之间的间隔，同样可以为负值。

3. mask-box-image

mask-box-image 定义遮罩图像，该图像将覆盖投影区域。遮罩图像可以是背景图片，也可以是渐变生成的背景图像。如果省略该值，则默认无遮罩图像。该参数有如下取值可供选择：

- (1) none：无遮罩。
- (2) 使用绝对或相对地址指定遮罩图像。
- (3) 使用线性渐变创建遮罩图像。
- (4) 使用径向（放射性）渐变创建遮罩图像。
- (5) 使用重复的线性渐变创建遮罩图像。
- (6) 使用重复的径向（放射性）渐变创建遮罩图像。

说明：

设置或检索对象倒影，对应的脚本特性为 box-reflect。

下面的实例定义一个简单的倒影样式，若我们想要为图 7.32 增加一个在水中的倒影，就需要利用 CSS3 的 box-reflect 特性来实现。

经我们使用 CSS3 为其添加 box-reflect 特性后，就形成了这两座山的倒影，效果如图 7.33 所示。



图 7.32 原图

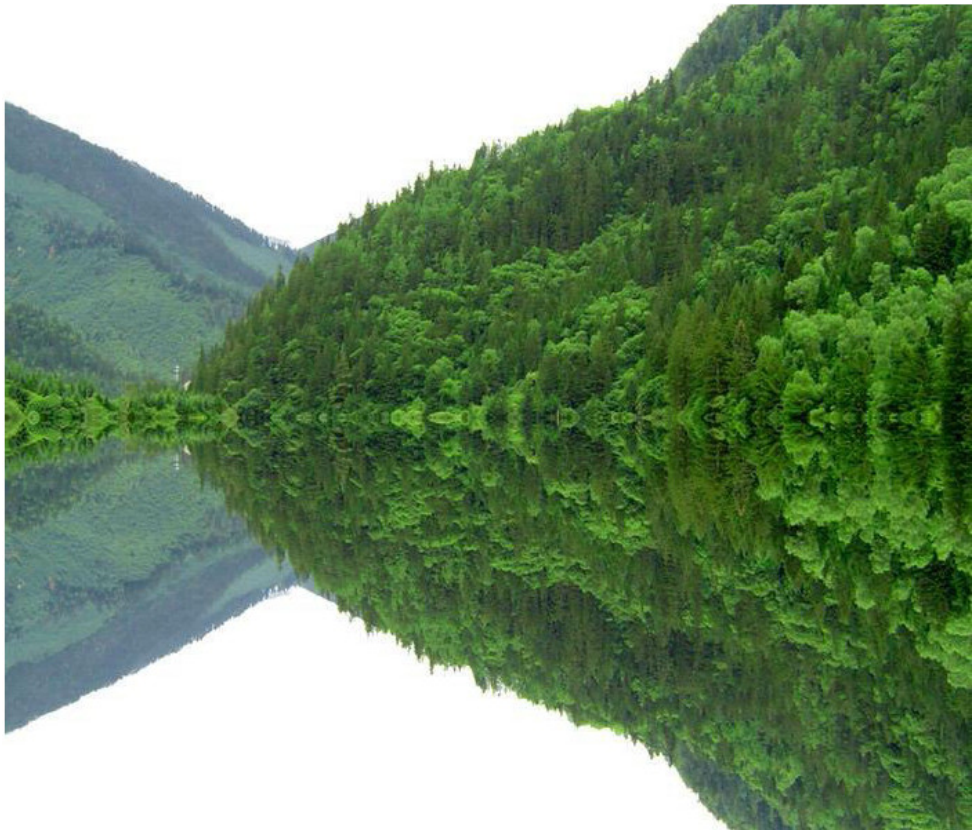


图 7.33 添加 box-reflect 特性后的效果



该实例的代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5     <title>倒影</title>
6     <style>
7       /* 为图片增加倒影，指定该倒影在其下面 */
8       img{ display:block;margin:20px auto;-webkit-box-reflect:below;}
9     </style>
10  </head>
11  <body>
12    
13  </body>
14 </html>
```



使用浏览器直接打开这个文件，就可以看到浏览器对这个网页文件解释后的结果，图片被加上了倒影，如图 7.34 所示。

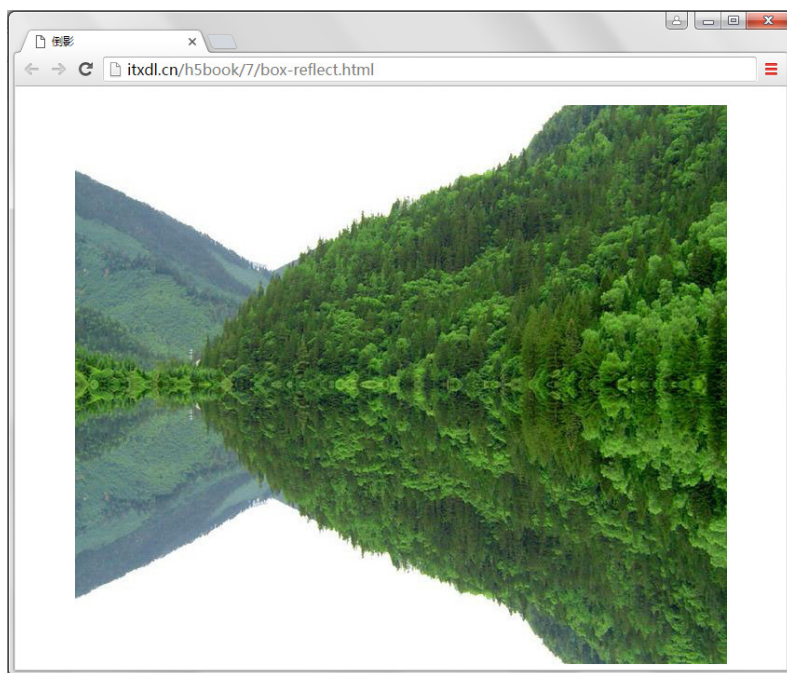


图 7.34 倒影

我们在上例倒影的基础上继续改进，为倒影设置距离，向下偏移 2px，倒影效果如图 7.35 所示。

修改的 CSS 代码如下：

```
6 <style>
7   /* 为图片增加倒影，指定该倒影在其下面，并向下偏移2px */
8   img{ display:block;margin:20px auto;-webkit-box-reflect:below 2px;}
9 </style>
```



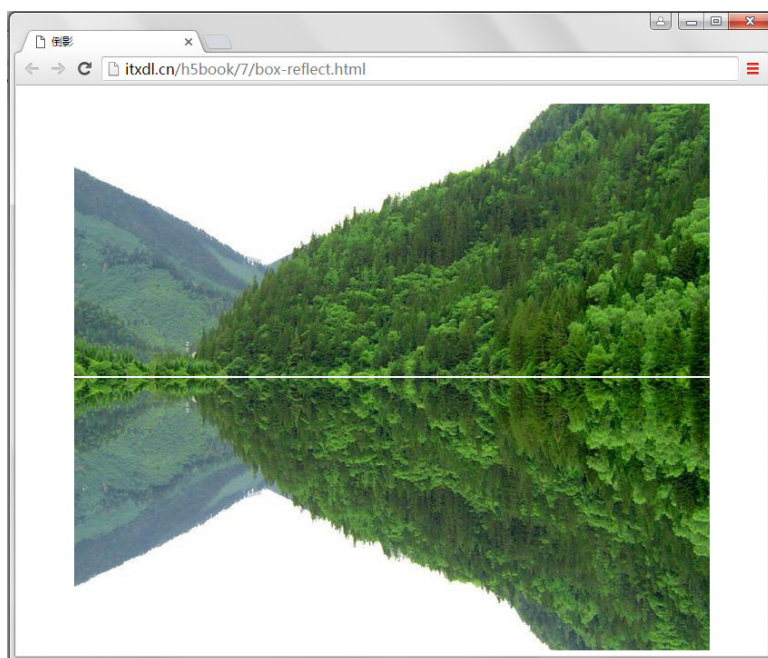


图 7.35 倒影向下偏移 2px

接下来继续进行改进，设计 CSS 渐变倒影，通过渐变遮罩逐渐盖住下面的倒影，制作出渐隐效果。效果图如图 7.36 所示。

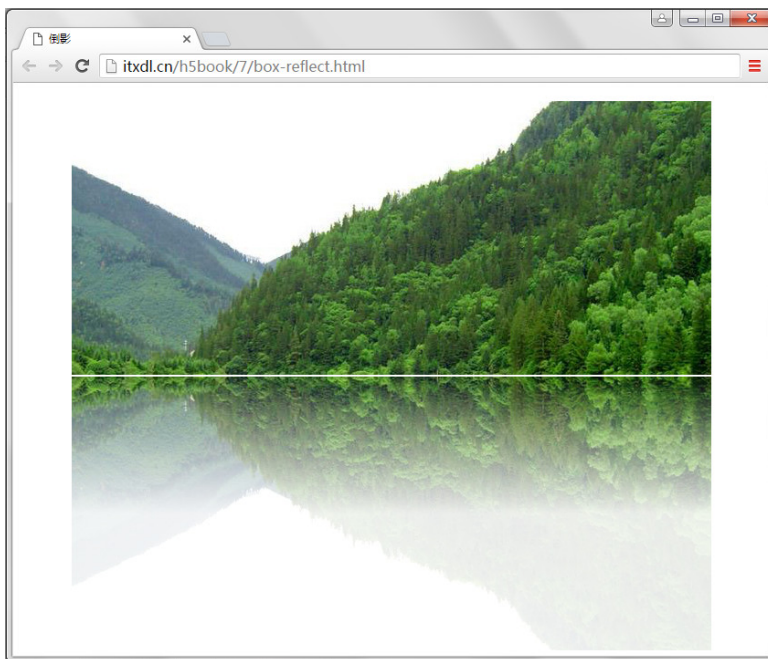


图 7.36 倒影渐隐效果



CSS 代码如下:

```
6 <style>
7 /* 为图片增加倒影, 指定该倒影向下偏移2px, 制作出渐隐效果 */
8 img{
9     display:block;margin:20px auto; -webkit-box-reflect:below 2px
10     -webkit-linear-gradient(bottom,rgba(0,0,0,1) 0,rgba(0,0,0,0.1) 50%);
11 }
12 </style>
```



另外, 我们除了可以为图片设计倒影, 网页上的任何对象都可以设计 CSS 倒影效果。下面的实例是将文本设计为倒影效果, 如图 7.37 所示。



图 7.37 文本倒影效果

文本倒影的 HTML 代码如下:

```
1 <!DOCTYPE>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>CSS3文字倒影效果</title>
6     <style>
7       #box{
8         width:500px; height:180px; margin:10px auto; border: 1px solid #000;
9         /* 为box添加倒影, 使得文本也出现倒影效果 */
10        -webkit-box-reflect: below 2px; text-align: center;
11      }
12    </style>
13  </head>
14  <body>
15    <div id="box">
16      <h2>兄弟连IT教育</h2>
17      <p>国内专业PHP培训学校、强大的PHP教学团队</p>
18      <p>培训PHP人才多、靠谱的企业合作</p>
19      <p>PHP培训校区大、学员就业牛</p>
20    </div>
21  </body>
22 </html>
```



倒影效果不会对其他元素产生影响，不会影响页面布局。当然，页面上的任何元素都可以用 CSS3 制作成倒影的形式，视频也不例外。

7.7 CSS3 分栏布局

CSS3 中新出现的多列布局（multi-column）是传统 HTML 网页中块状布局模式的有力扩充。这种新语法能够让 Web 开发人员轻松实现让文本呈现多列显示。我们知道，当一行文字太长时，读者读起来会比较费劲，有可能读错行或读串行；人们的视点从文本的一端移到另一端，然后换到下一行的行首，如果眼球移动浮动过大，则注意力就会减退，容易读不下去。所以，为了最大效率地使用大屏幕显示器，页面设计中需要限制文本的宽度，让文本按多列呈现，就像报纸上的新闻排版一样。但是在 CSS3 的多列布局（columns）语法功能出现之前，人们如果想让文本呈多列显示，要么使用绝对定位，手动给文本分段落，要么使用 JS 脚本等，而新语法的出现，彻底改变了这样的局面。

对于一些不支持多列布局特征的浏览器，比如 IE 9、IE 8，会把这些属性全部忽略，从而使布局呈现出传统的单块布局。为了保证浏览器最大的兼容性，我们在使用多列布局属性时，最好添加浏览器引擎前缀，最基本的要加上 3 种：谷歌浏览器的-webkit-，火狐浏览器的-moz-，IE 浏览器的-ms-。最后，别忘了不带前缀的写法。

分栏布局有 4 个参数可以设定，分别为 column-width、column-count、column-gap、column-rule，具体描述如表 7.11 所示。

表 7.11 CSS3 分栏布局的参数说明

参 数	描 述
column-width	栏目宽度。定义了宽度，元素会随着页面宽度的增加而增加列数
column-count	栏目标数。如果定义了列数，元素会随着页面宽度的增加而增加列宽度
column-gap	栏目距离
column-rule	栏目间隔线。和定义 border 一样

7.7.1 列个数和列宽度

不管想让一段文本呈多少列显示，你需要的只有两个属性：column-count 和 column-width。

（1）使用 column-count 将某个文章设置为两列，代码如下：



```
1 <!doctype html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>CSS3分栏布局</title>
6     <style>
7         #wrapper{
8             /* 为wrapper设置双栏布局, 每栏的宽度自动变为一半 */
9             width:100%;border:1px solid #000; font:14px/28px "宋体";color:#000; text-indent:2em;
10            -webkit-column-count:2;-moz-column-count:2; -ms-column-count:2; column-count:2;
11        }
12    </style>
13 </head>
14 <body>
15     <div id="wrapper">
16         兄弟连IT教育隶属于易第优(北京)教育咨询股份有限公司, 成立于2006年(
17         以下简称兄弟连)。专注于IT技术培训, 是国内专业的PHP/LAMP技术专业培
18         训学校。在兄弟连, 你可以找到自我、重拾自信; 在兄弟连, 你会每天渴求成长,
19         学到深夜; 在兄弟连, 你把学习当成一种习惯; 在兄弟连, 你有更多的兄弟姐
20         妹; 在兄弟连, 有陪你一起熬夜的老师; 在兄弟连, 你会被“狠狠”地爱着……兄
21         弟连已分别在北京、上海、广州、沈阳、郑州、济南、成都、杭州、南京、南
22         宁、深圳、天津等地设立校区, 每年有数十万名IT爱好者及从业人员受益于兄
23         弟连的职业培训、教学视频、网络公开课、院校讲座、出版书籍。“我们不仅仅
24         是老师, 我们是学员的梦想守护者与职场引路人。”
25     </div>
26 </body>
27 </html>
```



这将会使文本里的内容显示成两列(首先你的浏览器要支持这种新语法, 比如火狐浏览器、谷歌浏览器、IE10+等), 在这里我们使用了 column-count 兼容浏览器, 使用浏览器打开这个 HTML 文件, 效果如图 7.38 所示。



图 7.38 column-count 分栏布局

(2) column-width 属性控制列的宽度。如果你没有提供 column-count 属性值, 那么浏览器会自主决定将文本分成合适的列数。将上例的 CSS 代码替换如下:

```
6 <style>
7 #wrapper{
8     /* 为wrapper做分栏, 每栏的宽度固定为10em */
9     width:100%;border:1px solid #000; font:14px/28px "宋体";color:#000; text-indent:2em;
10    -webkit-column-width:10em;-moz-column-width:10em;-ms-column-width:10em;column-width:10em;
11 }
12 </style>
```



这使得文本里的内容被分成若干列，每列的宽度为 10em，分栏效果如图 7.39 所示。



图 7.39 column-width 分栏布局

7.7.2 列之间的缝隙间隔宽度

两列之间会有缝隙间隔。默认情况下这个间隔宽度是 1em，但如果使用 column-gap 属性，就会修改这个默认的宽度值。将上例的分栏宽度设置为 20em，分栏间隔设置为 2em。将上例的 CSS 代码替换如下：



```

6  <style>
7  #wrapper{
8      /* 为wrapper做分栏，每栏的宽度固定为20em，间隔宽度为2em */
9      width:100%;border:1px solid #000;font:14px/28px "宋体";color:#000;text-indent:2em;
10     -webkit-column-width:20em;-moz-column-width:20em;-ms-column-width:20em;column-width:20em;
11     -webkit-column-gap: 2em;-moz-column-gap: 2em;-ms-column-gap: 2em;column-gap: 2em;
12 }
13 </style>

```

这使得文本里的内容被分成若干列，每列的宽度为 20em，分栏间隔为 2em，效果如图 7.40 所示。

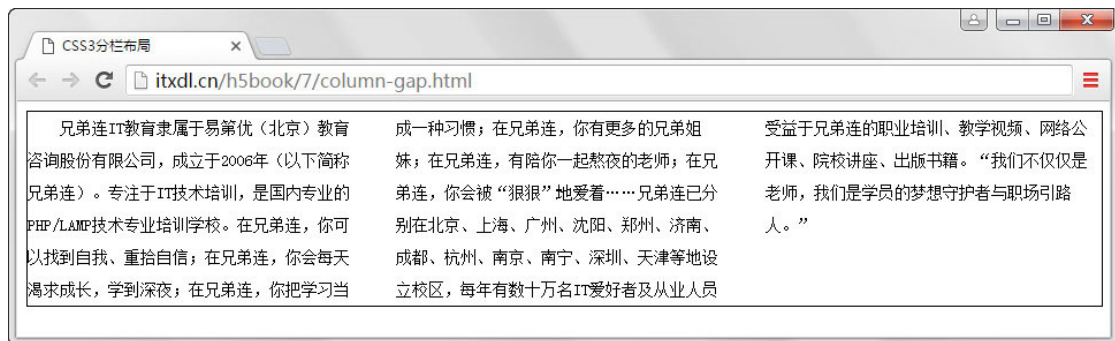


图 7.40 column-gap 分栏间隔



7.7.3 分栏间隔符

column-rule 的语法格式如下：

```
column-rule: column-rule-width column-rule-style column-rule-color;
```

column-rule 属性是一个简写属性，用于设置所有 column-rule-* 属性。它们的参数及描述如表 7.12 所示。

表 7.12 column-rule 的参数取值及描述

参 数	描 述
column-rule-width	设置列之间的宽度规则
column-rule-style	设置列之间的样式规则
column-rule-color	设置列之间的颜色规则

其中，column-rule-style 有以下可选值，它们的取值及描述如表 7.13 所示。

```
column-rule-style: none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset;
```

表 7.13 column-rule-style 参数取值及描述

取 值	描 述
none	定义没有规则
hidden	定义隐藏规则
dotted	定义点状规则
dashed	定义虚线规则
solid	定义实线规则
double	定义双线规则
groove	定义 3D grooved 规则。该效果取决于宽度和颜色值
ridge	定义 3D ridged 规则。该效果取决于宽度和颜色值
inset	定义 3D inset 规则。该效果取决于宽度和颜色值
outset	定义 3D outset 规则。该效果取决于宽度和颜色值

下面，我们将文本分为三栏，并给出 2em 的缝隙间隔，每栏间隔都有一个黑色的虚线间隔符。CSS 代码如下：

```

6 <style>
7 #wrapper{
8     /* 将wrapper分为3栏, 间隔宽度为2em, 间隔符为黑色虚线 */
9     width:100%;font:14px/28px "宋体";color:#000; text-indent:2em;
10    -webkit-column-count:3;-moz-column-count:3;-ms-column-count:3;column-count:3;
11    -webkit-column-gap: 2em;-moz-column-gap: 2em;-ms-column-gap: 2em;column-gap: 2em;
12    -webkit-column-rule: 3px dashed #000;-moz-column-rule: 3px dashed #000;
13    -ms-column-rule: 3px dashed #000;column-rule: 3px dashed #000;
14 }
15 </style>

```



这使得文本里的内容被分为三列, 分栏间隔为 2em, 分隔符为黑色虚线, 效果如图 7.41 所示。



图 7.41 column-rule 分栏间隔符

CSS3 的多列布局 (columns) 是一种方便 Web 开发者高效利用宽屏显示器的非常有用的功能特征。你会发现在很多地方都需要用到它们, 特别是在需要自动平衡列高度的地方。

7.8 圆角

传统的圆角生成方案, 必须使用多张图片作为背景图案。CSS3 的出现, 使得我们再也不必浪费时间去制作这些图片了, 只需要 border-radius 属性即可, 支持浏览器 IE 9、Opera 10.5、Safari 5、Chrome 4 和 Firefox 4。

7.8.1 border-radius属性

CSS3 圆角只需设置一个属性: border-radius (含义是“边框半径”)。为这个属性提供一个值, 就能同时设置四个圆角的半径。所有合法的 CSS 度量值都可以使用: em、px、百分比等。比如, 下面是一个 div 方框 (宽、高都是 200px, 背景为墨绿色, 边框为 2px 的灰色实线), 代码如下:



```
1 <!doctype html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>圆角</title>
6 <style>
7   /* 产生一个长、宽为200px的矩形 */
8   #box{ width:200px;height:200px;margin:0 auto;background:#009688;border:2px solid #212121;}
9 </style>
10 </head>
11 <body>
12   <div id="box"></div>
13 </body>
14 </html>
```



实现的效果如图 7.42 所示。



图 7.42 box 最初样式

现在来为它设置 50%的圆角，为 CSS 增加 border-radius: 50%，CSS 代码如下：

```
6 <style>
7 #box{
8   width:200px;height:200px;margin:0 auto;background:#009688;border:2px solid #212121;
9   /* 为矩形设置50%的圆角 */
10  border-radius: 50%;
11 }
12 </style>
```



这里将圆角属性值设置为 50%，也就是说，同时将每个圆角的“水平半径”和“垂直半径”都设置为 50%，最后实现一个圆形，如图 7.43 所示。

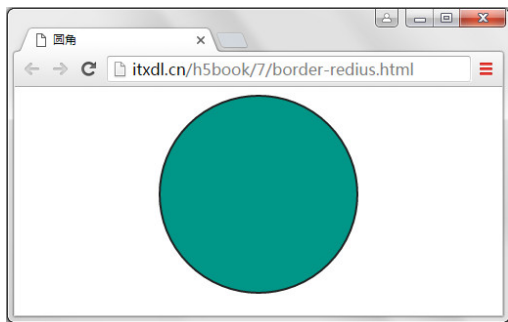


图 7.43 为 box 设定四角都为 50%的圆角

`border-radius` 可以同时设置 1~4 个值（想想我们之前的 `margin` 与 `padding`）。如果设置 1 个值，表示 4 个圆角都使用这个值。如果设置两个值，表示左上角和右下角使用第一个值，右上角和左下角使用第二个值。如果设置三个值，表示左上角使用第一个值，右上角和左下角使用第二个值，右下角使用第三个值。如果设置四个值，则依次对应左上角、右上角、右下角、左下角（顺时针顺序）。

现在我们来为 `box` 的 `border-radius` 属性设置两个值 `border-radius: 50px 25px`，来看看它的实现效果，CSS 代码如下：

```

6  <style>
7  #box{
8      width:200px;height:200px;margin:0 auto;background:#009688;border:2px solid #212121;
9      /* 表示左上角和右下角使用第一个值，右上角和左下角使用第二个值 */
10     border-radius: 50px 25px;
11 }
12 </style>

```



实现的圆角效果，将 `box` 的左上角和右下角的圆角半径设为 50px，右上角和左下角的圆角半径设为 25px，如图 7.44 所示。



图 7.44 为 `box` 的 `border-radius` 属性设置两个值

接着为 `box` 的 `border-radius` 属性设置三个值 `border-radius: 50px 10px 25px`，来看看它的实现效果，CSS 代码如下：

```

6  <style>
7  #box{
8      width:200px;height:200px;margin:0 auto;background:#009688;border:2px solid #212121;
9      /* 左上角使用第一个值，右上角和左下角使用第二个值，右下角使用第三个值 */
10     border-radius: 50px 10px 25px;
11 }
12 </style>

```



实现的圆角效果，将 `box` 的左上角的圆角半径设为 50px，右上角和左下角的圆角半径设为 10px，右下角的圆角半径设为 25px，如图 7.45 所示。



图 7.45 为 box 的 border-radius 属性设置三个值

最后我们为 box 的 border-radius 属性设置四个值 border-radius: 50px 10px 25px 0，来看看它的实现效果，CSS 代码如下：

```
6 <style>
7 #box{
8   width:200px;height:200px;margin:0 auto;background:#009688;border:2px solid #212121;
9   /* border-radius 设置4个值，分别是左上角，右上角，右下角，左下角（顺时针） */
10  border-radius: 50px 10px 25px 0;
11 }
12 </style>
```



实现的圆角效果，将 box 的左上角的圆角半径设为 50px，右上角的圆角半径设为 10px，左下角的圆角半径设为 25px，右下角的圆角半径设为 0，如图 7.46 所示。



图 7.46 为 box 的 border-radius 属性设置四个值

7.8.2 单个圆角的设置

除了同时设置四个圆角，还可以单独对每个角进行设置。对应四个角，CSS3 提供四个单独的属性：

- (1) border-top-left-radius。

- (2) border-top-right-radius。
- (3) border-bottom-right-radius。
- (4) border-bottom-left-radius。

这四个属性都可以同时设置 1~2 个值。如果设置 1 个值，则表示水平半径与垂直半径相等。如果设置 2 个值，则第一个值表示水平半径，第二个值表示垂直半径。

现在，我们单独为 box 的左上角设定 50px 的圆角，来看看它的实现效果，CSS 代码如下：

```

6  <style>
7  #box{
8      width:200px;height:200px;margin:0 auto;background:#009688;border:2px solid #212121;
9      /* 为box左上角设定50px的圆角半径 */
10     border-top-left-radius: 50px;
11 }
12 </style>

```



实现的圆角效果，将 box 的左上角的圆角半径设为 50px，如图 7.47 所示。

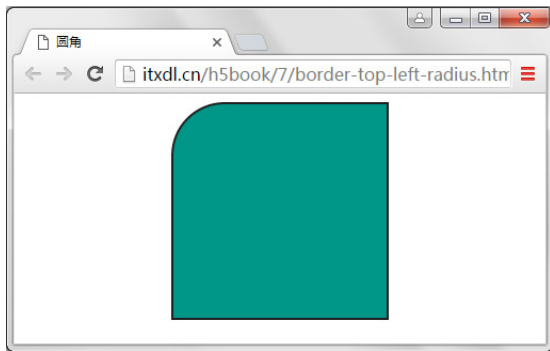


图 7.47 为 box 设置左上角的圆角

虽然各大浏览器都支持 border-radius，但是在某些细节上，实现并不一样。当四个角的颜色、宽度、风格（实线框、虚线框等）、单位都相同时，所有浏览器的渲染结果基本一致；一旦四个角的设置不相同，就会出现很大的差异。另外，并非所有浏览器都支持将圆角半径设为一个百分比值。

因此，目前最安全的做法就是，将每个圆角边框的风格和宽度都设为相同的值，并且避免使用百分比值。

7.9 边框

通过 CSS3，我们能够创建圆角边框，向矩形添加阴影，使用图片来绘制边框，并且不需要使用设计软件，比如 Photoshop（PS）。

7.9.1 边框图片border-image

border-image 为边框应用图片，顾名思义就是为图片应用背景图片。它和我们常用的 background 属性比较相似，但又比背景图片复杂一些。例如：

```
background: url("image.jpg") 10px 20px no-repeat;
```

border-image 属性是一个简写属性，用于设置以下属性，具体描述如表 7.14 所示。

表 7.14 CSS3 border-image 属性说明

属 性	描 述
border-image-source	用在边框的图片的路径
border-image-slice	图片边框向内偏移
border-image-width	图片边框的宽度
border-image-outset	边框图像区域超出边框的量
border-image-repeat	图像边框是否应平铺（repeated）、铺满（rounded）或拉伸（stretched）

想象一下：一个矩形，有四个边框。如果应用了边框图片，那么图片该怎么分布呢？图片会自动被分隔成四等分，用于四个边框。

可以把 border-image 理解成一个切片工具，它会自动对用作边框的图片进行切割。怎么切割呢？为了方便理解，下面有一张特殊的图片，它由 9 个正方形（对角线的长度为 70 像素）拼成（210×210 像素），并标注了序号，就如传说中的九宫格，如图 7.48 所示。



图 7.48 由 9 个正方形拼成的图片

将该图片应用于一个矩形的边框，并将边框图片的延伸方式设为重复，代码如下：

```

1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>边框图片</title>
6   <style>
7     #box{
8       width:210px; height:210px; margin: 0 auto; border:70px solid #ddd; background:#fff;
9       /* 边框图片地址border.jpg, 宽度为70px, 图片延伸方式是repeat(重复) */
10      border-image: url(border.jpg) 70 repeat;
11    }
12  </style>
13 </head>
14 <body>
15   <div id="box"></div>
16 </body>
17 </html>

```



运行上面的代码，我们可以看到该矩形的边框变成了图片，如图 7.49 所示。

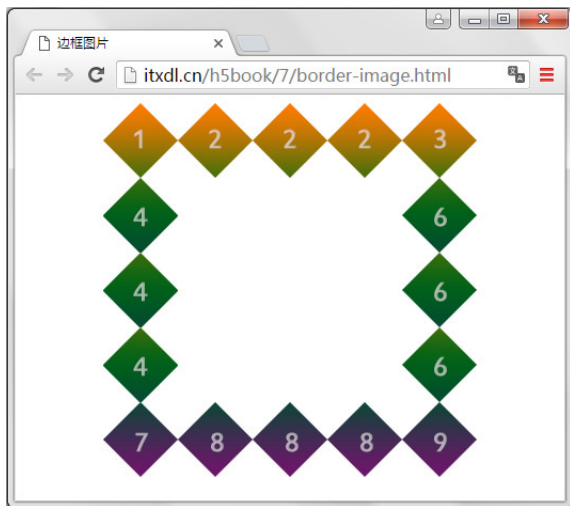


图 7.49 图片边框

从序号可以看出，div 的四个角分别对应了背景图片的四个角（1、3、9、7）。而 2、4、6、8 被重复在四个边。因为是从四周向中心切割图片的，所以 5 不显示。在上述例子中，border-image 包含三个参数，第一个参数“url(border.jpg)”表示边框图片的引入路径；第二个参数“70”表示切割图片的宽度，单位为像素，但一般省略单位，也可以使用百分比，当四边宽度相等时可以使用一个值，当设置四个值时遵循顺时针的规律来分别设置；第三个参数“repeat”表示图片的延伸方式为重复，有三个可选参数，即 round（平铺）、repeat（重复）、stretch（拉伸）。

repeat 的意思就是重复，因为上述例子中的边框宽度正好是字块宽度，所以效果不能很好地体现出来。下面改变 div 的宽度，再来看看 repeat 的效果，修改后 CSS 的样式如下：



```
6 <style>
7 #box{
8   width:180px; height:180px; margin: 0 auto; border:70px solid #ddd; background:#fff;
9   /* 边框图片地址border.jpg, 宽度为70px, 图片延伸方式是重复 */
10  border-image: url(border.jpg) 70 repeat;
11 }
12 </style>
```



这里我们将 div 的宽度缩小了 30px，变为 180px，在浏览器中查看的效果如图 7.50 所示。

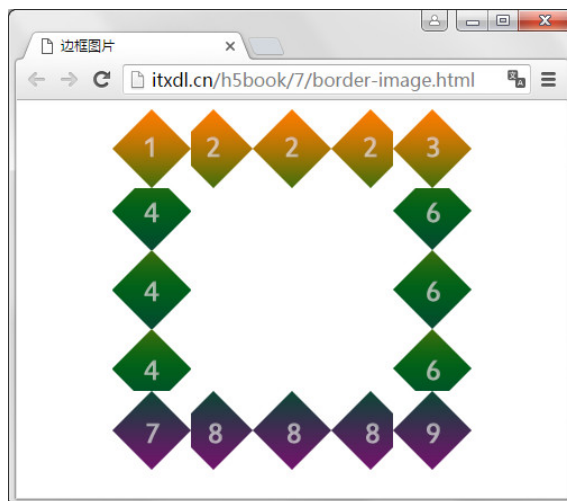


图 7.50 边框图片重复 (repeat)

在图 7.50 中，我们可以看出矩形的边角部分被裁掉了，repeat 就是从边框的中间一直重复并且超出部分被裁掉。

再来为 div 的 border-image-repeat 设置 round。round 可以理解为圆满地铺满，为了实现圆满，所以会压缩（或拉伸）。将上述例子的延伸方式改变，修改的 CSS 代码如下：

```
6 <style>
7 #box{
8   width:180px; height:180px; margin:0 auto; border:70px solid #ddd; background:#fff;
9   /* 边框图片地址border.jpg, 宽度为70px, 图片延伸方式是round(平铺) */
10  border-image: url(border.jpg) 70 round;
11 }
12 </style>
```



这里我们将边框图片的延伸方式修改为 round，在浏览器中查看的效果如图 7.51 所示。

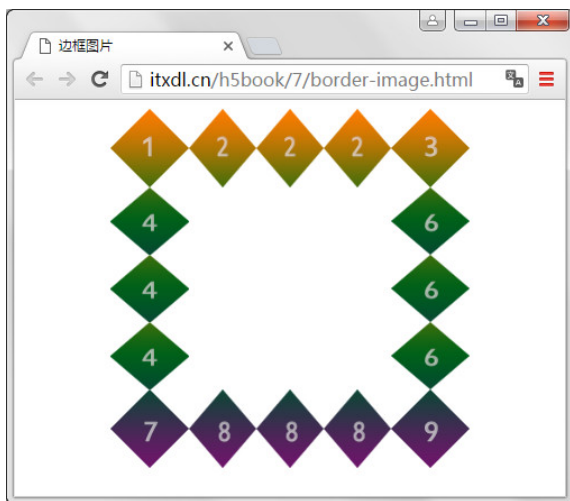


图 7.51 边框图片平铺 (round)

对比 repeat 的效果，我们可以发现 round 的图片被压缩了，并没有被截取。

接下来为 div 的 border-image-repeat 设置 stretch。stretch 为拉伸，将上述例子的延伸方式改变，修改的 CSS 代码如下：

```

6  <style>
7  #box{
8      width:180px; height:180px; margin:0 auto; border:70px solid #ddd; background:#fff;
9      /* 边框图片地址border.jpg，宽度为70px，图片延伸方式是stretch(拉伸) */
10     border-image: url(border.jpg) 70 stretch;
11  }
12 </style>

```



将 CSS 替换后，运行上述代码，可以发现边框图片被拉伸，而不是重复，在浏览器中查看的效果如图 7.52 所示。

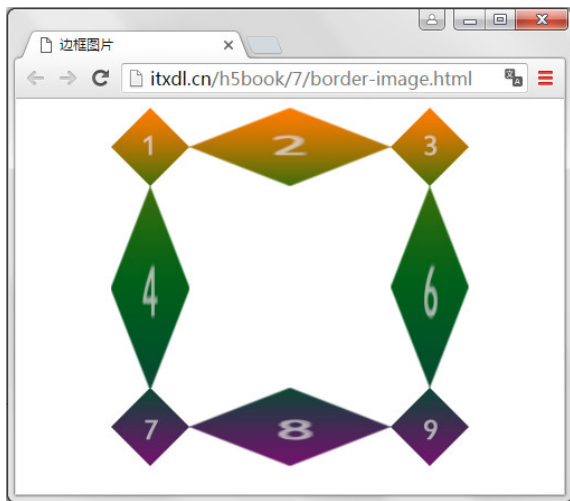


图 7.52 边框图片拉伸 (stretch)



7.9.2 自适应的圆角效果

实现该效果时使用的图片如下：



此图片的剪裁宽度为 20 像素，基本上就是此图片的圆角大小。div 的边框宽度为 10 像素，代码如下：

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>自适应的圆角效果</title>
6   <style>
7     #box{
8       width:200px; height:200px; border:10px solid; font-size:20px;
9       text-align:center; line-height:200px; margin:20px auto;
10      /* 图片的剪裁宽度为20像素，div的边框为10像素 */
11      border-image:url(border.png) 20/10px;
12    }
13  </style>
14 </head>
15 <body>
16   <div id="box">自适应圆角效果</div>
17 </body>
18 </html>
```



运行上述代码，div 的边框实现了上面图片制作的圆角效果。在浏览器中进行查看，效果如图 7.53 所示。



图 7.53 自适应圆角效果

同样，我们还可以使用图片来实现多边框效果、投影效果和选项卡效果等。

1. 多边框效果

使用下面图片来实现多边框效果：



使用上面的代码，修改边框图片，在浏览器中进行查看，效果如图 7.54 所示。



图 7.54 多边框效果

2. 投影效果

使用下面图片来实现投影效果，投影可以控制剪裁宽度和边框宽度。



使用 border-image 实现的效果如图 7.55 所示。



图 7.55 投影效果



`border-image` 可以说是 CSS3 中非常重要的部分，将来一定会大放光彩，其应用潜力非常惊人，但目前支持的浏览器有限，仅 Firefox、Chrome、Safari 3+ 浏览器支持。

7.10 渐变

渐变背景在 Web 页面中是一种常见的视觉元素，但一直以来，Web 设计师都是通过图形软件设计这些渐变效果，然后以图片形式或者背景图片的形式运用到页面中。Web 页面上实现的效果，仅从页面的视觉效果上来看，与设计并无任何差异。事实上，这种方法比较麻烦，因为首先需要设计师进行设计，然后进行切图，再通过样式应用到页面中。另外，在实际应用中可扩展性差，还直接影响页面性能。值得庆幸的是，W3C 组织将渐变设计收入 CSS3 标准中，让广大的前端设计师直接受益，可以直接通过 CSS3 的渐变属性制作类似渐变图片的效果。而且渐变属性渐渐得到了众多现代浏览器的兼容，甚至是 IE，IE 10 版本也支持这个属性。

CSS3 gradient 分为 `linear-gradient`（线性渐变）和 `radial-gradient`（径向渐变）。

7.10.1 CSS3 渐变介绍

欲要了解 CSS3 渐变，就先要知道 CSS3 渐变是什么。从早前的设计中我们可以知道，渐变是两种或多种颜色之间的平滑过渡。在创建渐变的过程中，可以指定多个中间颜色值，这个值称为色标。每个色标包含一种颜色和一个位置，浏览器从每个色标的颜色淡出到下一个，以创建平滑的渐变，如图 7.56 所示。

渐变可以应用于任何使用背景图片的地方。这意味着在 CSS 样式中，渐变相当于背景图片，在理论上可在任何使用 `url()` 值的地方采用，比如常见的 `background-image`、`list-style-type` 以及前面介绍的 CSS3 的图像边框属性 `border-image`。但直到目前为止，仅在背景图片中得到完美的支持。



图 7.56 渐变编辑器

7.10.2 线性渐变

在线性渐变的过程中，颜色沿着一条直线过渡：从左侧到右侧、从右侧到左侧、从顶部到底部、从底部到顶部或沿任意轴。如果你曾使用过图形制作软件，比如 Photoshop，那么对线性渐变就并不陌生。

CSS3 制作渐变效果，其实和使用制作软件中的渐变工具没有什么差别。首先需要指定渐变的方向、渐变的起始颜色、渐变的结束颜色。具备这三个参数，就可以制作一个简单、普通的渐变效果。如果你需要制作一个复杂的多色渐变效果，就需要在同一个渐变方向增添多个色标。具备这些渐变参数（至少三个），各浏览器就会绘制与渐变线垂直的颜色结来填充整个容器。浏览器渲染出来的效果就类似于制作软件设计出来的渐变图像，从一种颜色到另一种颜色的平滑淡出，沿所指的线性渐变方向实现颜色渐变效果。

线性渐变的格式如下：

```
linear-gradient([<起点> || <角度>,<点>,<点>...)
```

线性渐变的参数描述如表 7.15 所示。

表 7.15 CSS3 线性渐变的参数描述

参 数	描 述
起点	从什么方向开始渐变（默认是 top）
角度	从什么角度开始渐变（xxxdeg 的格式）
点	渐变点的颜色和位置

7.10.3 线性渐变实例

1. 颜色从顶部向底部渐变

制作从顶部向底部直线渐变有三种方法，第一种方法是起点参数不设置，因为起点参数的默认值为“top”；第二种方法是起点参数设置为“top”；第三种方法是起点参数使用“-90deg”关键词。为 top_bottom 设置从顶部向底部的渐变，三种方法的 CSS 代码如下。

第一种方法：

```
6 <style>
7 .top_bottom{
8     width: 200px; height: 200px; margin:10px auto;
9     /* 线性渐变的起点参数不设置 */
10    background-image: -webkit-linear-gradient(white,black);
11    background-image: linear-gradient(white,black);
```





```
12 }  
13 </style>
```

第二种方法:

```
6 <style>  
7 .top_bottom{  
8     width: 200px; height: 200px; margin:10px auto;  
9     /* 线性渐变的起点参数设置为top */  
10    background-image: -webkit-linear-gradient(top,white,black);  
11    background-image: linear-gradient(top,white,black);  
12 }  
13 </style>
```



第三种方法:

```
6 <style>  
7 .top_bottom{  
8     width: 200px; height: 200px; margin:10px auto;  
9     /* 线性渐变的起点参数设置为-90deg */  
10    background-image: -webkit-linear-gradient(-90deg,white,black);  
11    background-image: -linear-gradient(-90deg,white,black);  
12 }  
13 </style>
```



上述三种 CSS 设置运行的效果相同, top_bottom 的背景变为从白色到黑色、自上而下形成线性渐变。在浏览器中进行查看, 效果如图 7.57 所示。

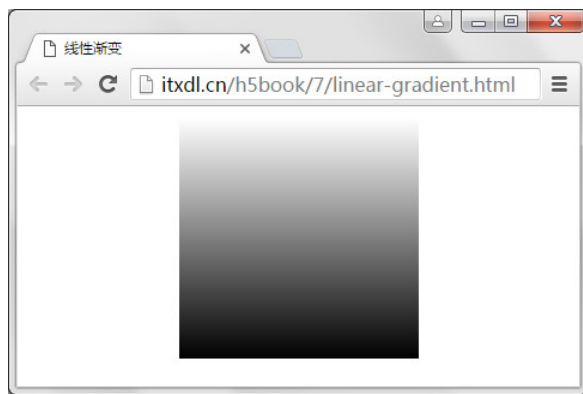


图 7.57 颜色从顶部向底部渐变

同样, 我们可以将起点参数设置为 bottom、left、right, 依次是从底部向顶部渐变、从左边向右边渐变和从右边向左边渐变。

2. 颜色从左下角向右上角渐变

制作从左下角向右上角直线渐变有两种方法, 第一种方法是起点参数设置为“bottom left”; 第二种方法是起点参数使用“45deg”关键词。为 left_bottom 设置从顶部向底部的渐变, CSS 代码如下。

第一种方法:

```
7 .left_bottom{
8   width: 200px; height: 200px; margin:20px auto;
9   background-image: -webkit-linear-gradient(bottom left,white,black);
10  background-image: -linear-gradient(bottom left,white,black);
11 }
```



第二种方法:

```
7 .left_bottom{
8   width: 200px;
9   height: 200px;
10  margin: 20px auto;
11  background-image: -webkit-linear-gradient(bottom left, white, black);
12  background-image: -linear-gradient(45 deg, white, black);
13 }
```



上述两种 CSS 设置运行的效果相同, left_bottom 的背景变为从左下角到右上角形成线性渐变。在浏览器中进行查看, 效果如图 7.58 所示。

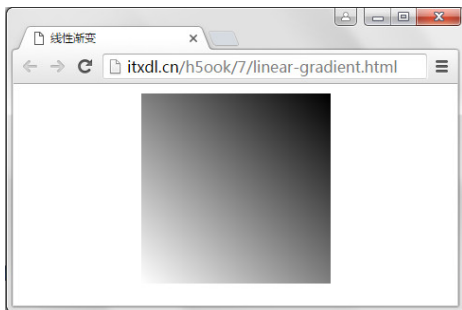


图 7.58 颜色从左下角向右上角渐变

3. 多色线性渐变

前面向大家演示的效果仅是一些简单的线性渐变(两色渐变), 其实在实际中, 渐变不仅仅只有两种颜色, 而是会有多种颜色。接下来, 我们一起来看一个从左向右的五彩渐变, 代码如下:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>五色渐变</title>
6   <style>
7     .to_left{
8       width: 800px; height: 200px; margin:20px auto;
9       background-image: -webkit-linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
10      background-image: -linear-gradient(left,red,orange,yellow,green,blue,indigo,violet);
11    }
12  </style>
13 </head>
14 <body>
15   <div class="to_left"></div>
16 </body>
17 </html>
```





将包含上述代码的 HTML 文件在浏览器中进行查看，渐变效果如图 7.59 所示。

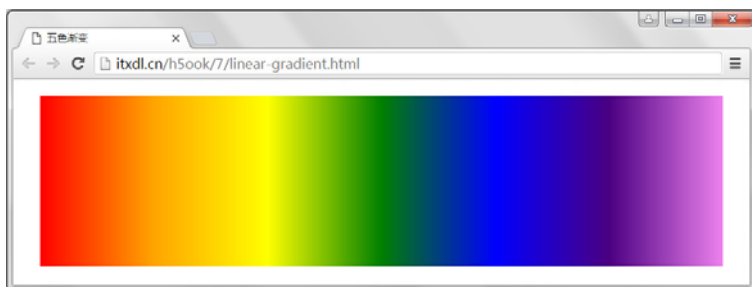


图 7.59 多色线性渐变

4. 自定义线性渐变

那么问题来了，如何实现图 7.60 的渐变效果？



图 7.60 自定义线性渐变（1）

通过图 7.60 可以注意到，颜色从 0% 的不透明度开始，第一个色标的位置位于 0%，其透明度为 0%；第二个色标的位置为 80% 的不透明度，位置位于 7%。我们可以通过渐变工具从 CSS 声明中捕捉相关数据，实现自定义线性渐变，CSS 代码如下：

```
7 .to_left{
8   background-image:-webkit-linear-gradient(
9     left,
10    rgba(255,0,0,0) 0%,
11    rgba(255,0,0,0.8) 7%,
12    rgba(255,0,0,1) 11%,
13    rgba(255,0,0,1) 12%,
14    rgba(255,252,0,1) 28%,
15    rgba(1,180,7,1) 45%,
16    rgba(0,234,255,1) 60%,
17    rgba(0,3,144,1) 75%,
18    rgba(255,0,198,1) 88%,
19    rgba(255,0,198,0.8) 93%,
20    rgba(255,0,198,0) 100%);
21  };
22  background-image:-linear-gradient(
23    left,
24    rgba(255,0,0,0) 0%,
25    rgba(255,0,0,0.8) 7%,
26    rgba(255,0,0,1) 11%,
27    rgba(255,0,0,1) 12%,
28    rgba(255,252,0,1) 28%,
29    rgba(1,180,7,1) 45%,
30    rgba(0,234,255,1) 60%,
31    rgba(0,3,144,1) 75%,
```



```

32     rgba(255,0,198,1) 88%,
33     rgba(255,0,198,0.8) 93%,
34     rgba(255,0,198,0) 100%);
35 );
36 }

```

将替换了 CSS 代码的 HTML 在浏览器中进行查看，渐变效果如图 7.61 所示。

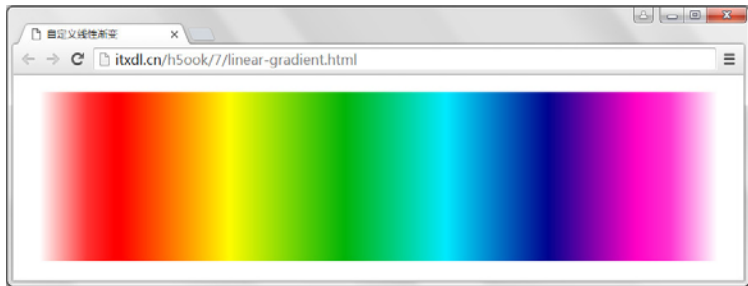


图 7.61 自定义线性渐变 (2)

图 7.61 证明，一个渐变可以包含多个色标，位置值为 0~1 的小数，或者 0~100% 的百分数，指定色标的位置比例，其用法与 Photoshop 中的直线渐变工具相似。

5. 进度条

应用 CSS3 线性渐变，我们还可以实现进度条效果。制作进度条，我们需要使用 `repeating-linear-gradient` 参数。`repeating-linear-gradient` 的含义是用重复的线性渐变创建图像。`repeating-linear-gradient()` 的语法与 `linear-gradient()` 相同。实现进度条效果，代码如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>进度条</title>
6 <style>
7     .wrap{
8         width:200px;height:30px;overflow:hidden;border:1px solid #000;
9     }
10    .box{
11        width:400px;height:30px;
12        background:repeating-linear-gradient(75deg,green 0,green 10px,#fff 10px,#fff 20px);
13        transition:3s; /* 过渡效果，下面的内容会提到 */
14    }
15    .wrap:hover .box{
16        margin-left:-100px; /* 鼠标移入wrap, box移动 */
17    }
18 </style>
19 </head>
20 <body>
21 <div class="wrap">
22     <div class="box"></div>
23 </div>
24 </body>
25 </html>

```



首先构造一个进度条，当鼠标移入进度条时，进度条开始运动，在浏览器中查看效果如图 7.62 所示。

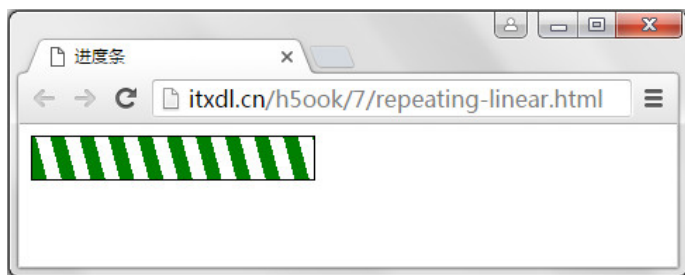


图 7.62 进度条

7.10.4 径向渐变

CSS3 径向渐变是圆形或椭圆形渐变。颜色不再沿着一条直线轴变化，而是从一个起点朝所有方向混合。但相对线性渐变，径向渐变要复杂得多。

径向渐变的格式如下：

```
radial-gradient([<起点>]? [<形状> || <大小>,<点>,<点>...])
```

径向渐变的参数描述如表 7.16 所示。

表 7.16 CSS3 径向渐变的参数描述

参 数	描 述
起点	可以是关键字（left、top、right、bottom）、具体数值或百分比
形状	椭圆 ellipse、圆形 circle
大小	具体数值或百分比，也可以是关键字[最近端、最近角、最远端、最远角、包含或覆盖（closest-side、closest-corner、farthest-side、farthest-corner、contain or cover）]

7.10.5 径向渐变实例

虽然径向渐变要比线性渐变更复杂，但只要了解了其基本语法以及相关属性参数的作用，并不需要花太多的时间去适应。接下来，我们通过实战来加强径向渐变的使用。本节的所有例子我们都在一个宽度为 400px，高为 300px 的容器中实现。

1. 从容器内部向外径向渐变

先来看一个简单的径向渐变，圆心都是容器正中间，从“#ffc107”颜色向“pink”颜色实现径向渐变，CSS 代码如下：

```
7 .box{  
8   width:400px;height:300px;margin:20px auto;  
9   background:-webkit-radial-gradient(#ffc107,pink);  
10  background:radial-gradient(#ffc107,pink);  
11 }
```



在浏览器中进行查看，效果如图 7.63 所示。

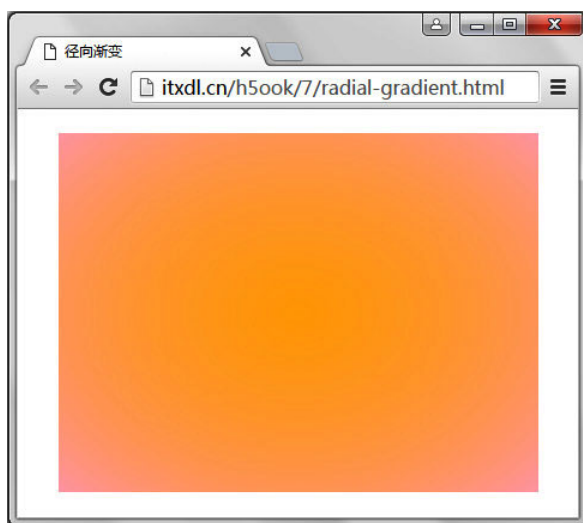


图 7.63 简单径向渐变

如果你想制作一个圆形渐变，而不是一个椭圆形渐变，则只需要添加一个关键词“circle”。我们在上例的基础上添加一个关键词“circle”，代码如下：

```
7 .box{  
8   width:400px;height:300px;margin:20px auto;  
9   background:-webkit-radial-gradient(circle,#ffc107,pink);  
10  background:radial-gradient(circle,#ffc107,pink);  
11 }
```



此时的渐变变成了圆形，在浏览器中查看效果如图 7.64 所示。

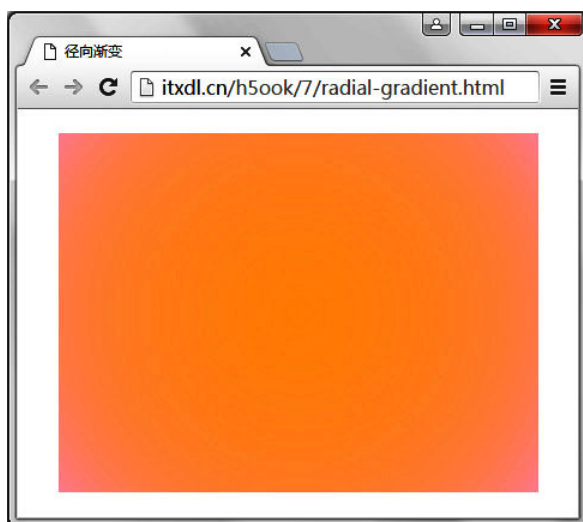


图 7.64 圆形渐变



正如你所看到的，圆形渐变是一个特殊的椭圆渐变，水平半径和垂直半径具有相同的长度值。既然圆形渐变是椭圆渐变的一种特殊情况，那么当渐变主要半径（水平半径）和次要半径（垂直半径）不相同，就是一个椭圆渐变。正如上面所言，主要半径和次要半径不相等时，制作的径向渐变是椭圆渐变，在制作椭圆渐变时，可以使用关键词“ellipse”。

2. 规定径向渐变的半径

除了使用关键词制作不同的径向渐变，还可以用不同的渐变参数制作径向渐变效果。通过制作同心圆、主要半径和次要半径，来决定径向渐变的形状。径向渐变的半径设为“200px,100px”。水平半径为 200px，垂直半径为 100px，从“#ffc107”色到“pink”色径向渐变，代码如下：

```
6 <style>
7 .box{
8     width:400px;height:300px;margin:20px auto;
9     background:-webkit-radial-gradient(200px 100px,#ffc107,pink);
10    background:radial-gradient(200px 100px,#ffc107,pink);
11 }
12 </style>
```



此时的渐变变成了水平半径为 200px、垂直半径为 100px 的椭圆，在浏览器中查看效果如图 7.65 所示。

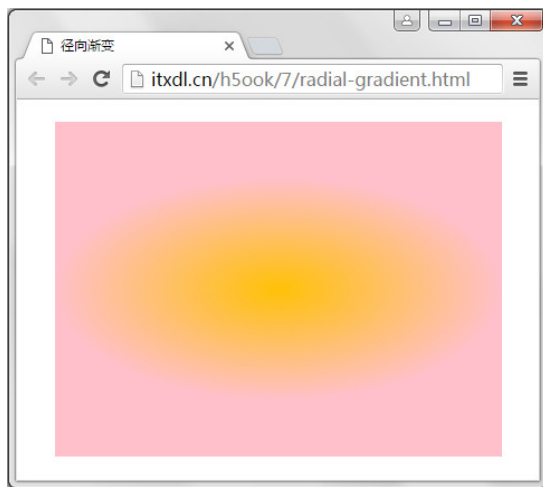


图 7.65 自定义半径的径向渐变

3. 规定径向渐变的半径及圆心位置

除了上述方法能实现一些简单的径向渐变效果，还可以使用渐变形状配合圆心定位。主要使用“at”加上关键词来定义径向渐变中心位置。径向渐变中心位置类似于 background-position 属性。例如，圆心位置在“100px,50px”处，水平半径为 200px，垂直半径为 100px，从“#ffc107”色到“pink”色径向渐变，代码如下：

```

6 <style>
7 .box{
8   width:400px;height:300px;margin:20px auto;
9   background:-webkit-radial-gradient(200px 100px at 100px 50px,#ffc107,pink);
10  background:radial-gradient(200px 100px at 100px 50px,#ffc107,pink);
11 }
12 </style>

```



此时的渐变变成了水平半径为 200px、垂直半径为 100px 且圆心位置在 “100px,50px” 处的椭圆，在浏览器中查看效果如图 7.66 所示。

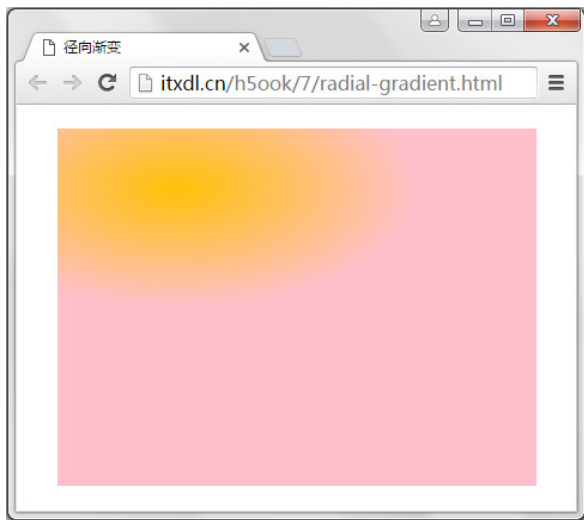


图 7.66 自定义半径及圆心位置的径向渐变

设置圆心位置除了可以使用特定的值，还可以使用百分比和一些关键词来定义，如 “center”、“top”、“right”、“bottom”、“left” 及这些关键词的组合，如 “top left”、“right bottom” 等，组合位置的关键词顺序可以调换。

4. 重复的径向渐变

跟线性渐变一样，我们也可以为径向渐变设置重复。以同样的方式，可以使用相关的属性创建重复的径向渐变。其语法和 `linear-gradient` 类似，只是以一个径向渐变为基础进行重复渐变，如下例所示，我们制作一个三色重复的径向渐变：

```

6 <style>
7 .box{
8   width:400px;height:300px;margin:20px auto;
9   background:repeating-radial-gradient(#E49EED,#fff 30px, pink 80px);
10 }
11 </style>

```



在浏览器中进行查看，渐变效果如图 7.67 所示。



图 7.67 三色重复的径向渐变

理解上述几个实例后，读者就可以 DIY 渐变效果了。

7.11 CSS3 背景

CSS3 规范中对背景这一部分，新加入了一些有用的功能，如可以设置多个背景图片、可以指定背景大小、设置背景渐变等功能。CSS3 规范中定义的背景属性新增了 background-clip、background-origin、background-size，这三个属性描述如表 7.17 所示。

表 7.17 CSS3 新增背景属性

属 性 名	取 值
background-clip	CSS3 新增属性, border-box padding-box 表示背景渲染的方法: padding box 表示背景在 padding box 内渲染; border-box 表示背景在 border-box 内渲染 (默认值为 border-box)
background-origin	CSS3 新增属性, border-box padding-box content-box 背景相对的位置, 相对于上面 3 个值中的一个 (默认值是 padding-box)
background-size	CSS3 新增属性, [length % auto]{1,2} cover contain 设置背景的大小。contain 表示按比例缩放占据最大高度或者宽度的背景; cover 表示铺满整个背景 (默认值是 auto)

7.11.1 多背景

CSS3 允许设置多个背景图片，每个背景用逗号隔开，每个背景图片占一层，层的上下显示按照在 CSS 中写的顺序来定，最先写的背景在最上层。

多背景的格式如下：

```
background: url(a.jpg) 0 0, url(b.jpg) 0 100%;
```

这里，我们使用三个图片为 div 制作三层背景，第一种方法是背景的简写方式，代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>多层背景</title>
6   <style>
7     #box{
8       width: 600px; height: 200px; margin: 0 auto;
9       background: url("../img/bg1.jpg") 0 0 no-repeat,
10        url("../img/itxdl.jpg") 200px 0 no-repeat,
11        url("../img/bg3.jpg") 400px 0 no-repeat;
12     }
13   </style>
14 </head>
15 <body>
16   <div id="box"></div>
17 </body>
18 </html>
```



上述代码为 div 设置了三层背景，每层的背景图片平铺方式设为不重复。第一层背景引用了“bg1.jpg”图片，位于该 div “0 0”的位置；第二层背景引用了“bg2.jpg”图片，位于该 div “200px 0”的位置；第三层背景引用了“bg3.jpg”图片，位于该 div “400px 0”的位置。在浏览器中运行的结果如图 7.68 所示。

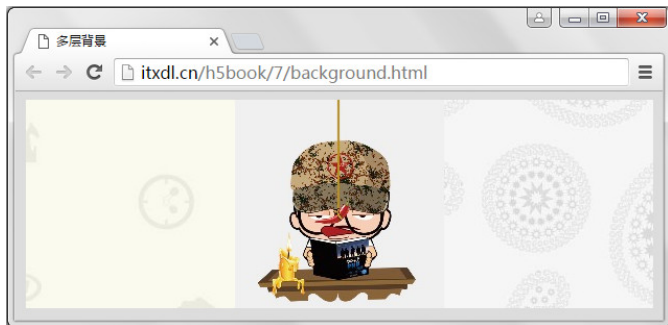


图 7.68 三层背景重叠

除了这种简写方法，我们还可以将 background 属性拆分开来，使用 background-image、background-repeat 和 background-position 分别来设置背景图片的地址、平铺方式和位置。CSS 代码如下：

```
6 <style>
7 body{ background: #ddd; }
8 #box{
9   width: 600px; height: 200px; margin: 0 auto;
```




```
10     background: url("../img/bg1.jpg"), url("../img/itxdl.jpg"), url("../img/bg3.jpg");
11     background-repeat: no-repeat, no-repeat, no-repeat;
12     background-position: 0 0, 200px 0, 400px 0;
13 }
14 </style>
```

这段代码是对上面代码的一个改写，在浏览器中的执行效果相同。值得注意的是，背景渐变也是一种背景层。下面我们为 div 设置两层背景，第一层背景为背景渐变，第二层背景为背景图。CSS 代码如下：

```
6 <style>
7 #box{
8     width: 400px; height: 200px; margin: 0 auto;
9     background-image: -webkit-linear-gradient(left, rgba(120, 120, 120, 1),
10         rgba(120, 0, 0, 0)), url("../img/itxdl.jpg");
11     background-repeat: no-repeat, no-repeat;
12     background-position: 0 0, 200px 0;
13 }
14 </style>
```



第一层为从左到右的“rgba(255,255,0,1)”到“rgba(255,0,0,0)”的渐变背景，第二层为“itxdl.jpg”的背景图。在浏览器中运行的结果如图 7.69 所示。

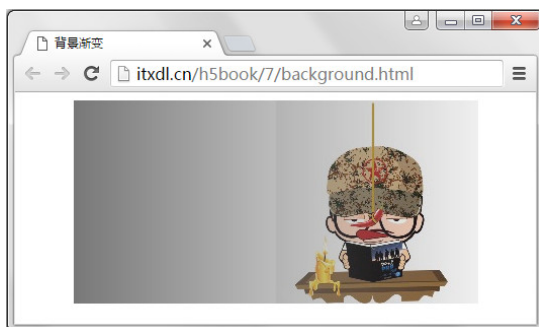


图 7.69 背景渐变层

7.11.2 background-size

CSS3 新增了背景尺寸 background-size 属性，该属性可以设置具体的值、百分比等。该属性的取值及描述如表 7.18 所示。

表 7.18 background-size 属性的取值及描述

属 性 名	取 值
length	设置背景图像的高度和宽度。第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为“auto”
百分比	以父元素的百分比来设置背景图像的宽度和高度。第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为“auto”

续表

属 性 名	取 值
cover	把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。背景图像的某些部分也许无法显示在背景定位区域中。倾向于放大背景图
contain	把图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域。倾向于缩小背景图

当我们为 `background-size` 设置一个值时，第二个值会被设置为“`auto`”。若为背景图片设置两个值时未与背景图本身大小成比例，则容易使图片失真或变形。

下面有一个高度和宽度为 `200px` 的 `div`，而我们想要将一张宽度和高度为 `600px` 的图片作为背景图。如果我们想要将该图片填充到这个 `div`，若没有 `background-size` 属性，则需要使用一些软件将这张图片压缩再设置为背景，否则背景图片会溢出。未使用 `background-size` 时，CSS 代码如下：

```
6 <style>
7 #box{
8     width: 200px; height: 200px; margin: 0 auto; border: 1px solid;
9     background: url("../img/itxd12.jpg");
10 }
11 </style>
```



在浏览器中运行的结果如图 7.70 所示。

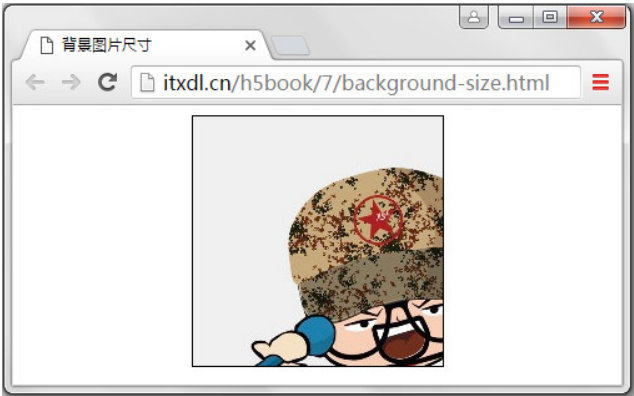


图 7.70 背景图片未使用 `background-size` 的效果

在这里我们只能看到“`itxd12.jpg`”的左上角的三分之一。将该背景图片的 `background-size` 设置为“`200px 200px`”、“`200px auto`”、“`auto 200px`”、“`cover`”和“`contain`”都可以达到全填充的目的。例如，我们在这里使用“`200px auto`”，增加“`background-size: 200px auto`”，CSS 代码如下：



```
6 <style>
7 #box{
8     width: 200px; height: 200px; margin: 0 auto; border: 1px solid;
9     background: url("../img/itxd12.jpg");
10    background-size: 200px auto;
11 }
12 </style>
```



在浏览器中运行的结果如图 7.71 所示。



图 7.71 背景图片全填充

只有当 background-size 的值为 auto 时，背景图片才不会变形或失真，而其他值都会造成背景图片变形或失真，所以大家使用时需要仔细考虑，以免给自己的制作带来不良效果。

7.11.3 background-origin

CSS3 可以使用 background-origin 属性设置图像的基准位置。该属性的取值及描述如表 7.19 所示。

表 7.19 background-origin 属性的取值及描述

属 性 名	取 值
padding-box	背景图像填充框的相对位置，并且是该属性的默认值
border-box	背景图像边界框的相对位置
content-box	背景图像的相对位置的内容框

该属性默认取值为“padding-box”，顾名思义，从图像的填充框开始绘制。该属性指定了背景从哪个区域（边框、补白或内容）开始绘制，但也仅仅能控制背景开始绘制的位置，你可以用这个属性在边框上绘制背景，但边框上的背景是否显示出来则由 background-clip 来决定。background-clip 在下面的内容中会提到。

这里，我们将 div 的边框设为半透明，体现出边框，又体现出边框中的背景。将其背景

图的 background-origin 设置为 border-box，将其边框设置为半透明。这时图像从边框界开始填充，代码如下：

```
6 <style>
7 #box{
8     width: 200px; height: 200px; margin: 0 auto; border: 10px solid rgba(0,0,0,0.5);
9     background: url("../img/itxd12.jpg") no-repeat;
10    background-size: 150px auto;
11    background-origin: border-box;
12 }
13 </style>
```



该 div 的背景图从边框开始绘制，在浏览器中运行的结果如图 7.72 所示。

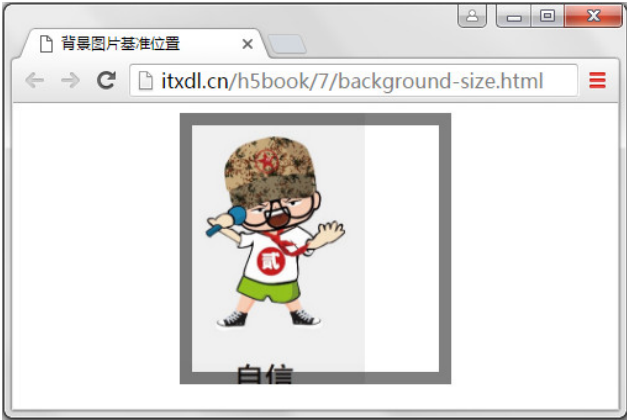


图 7.72 背景图片基准位置

7.11.4 background-clip

CSS3 可以使用 background-origin 属性规定背景的绘制区域。background-clip 控制的不仅仅是背景图片，而且还控制整个元素背景的显示范围。该属性的取值及描述如表 7.20 所示。

表 7.20 background-origin 属性的取值及描述

属性名	取 值
border-box	背景被裁剪到边框盒。并且是该属性的默认值
padding-box	背景被裁剪到内边距框
content-box	背景被裁剪到内容框
no-clip	从 border 区域向外裁剪背景

该属性的默认取值为“border-box”。该属性指定了背景在哪些区域可以显示，但与背景开始绘制的位置无关，背景绘制的位置可以出现在不显示背景的区域，这时就相当于背景图



片被不显示背景的区域裁剪了一部分。

这里，为 div 设置“padding: 10px”。将其背景图的 background-clip 设置为 content-box。这时图像背景从边框界开始填充，CSS 代码如下：

```
6 <style>
7 #box{
8   width: 200px; height: 200px; margin: 0 auto; border: 1px solid #000; padding: 10px;
9   background: url("../img/itxd12.jpg") no-repeat;
10  background-size: 150px auto;
11  background-clip: content-box;
12 }
13 </style>
```



在浏览器中运行的结果如图 7.73 所示。

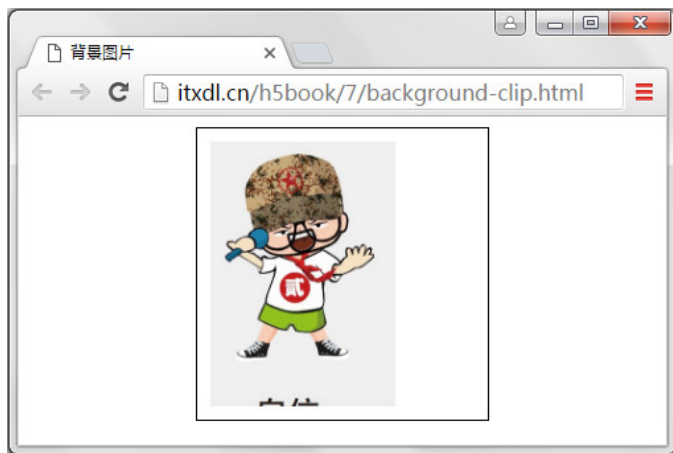


图 7.73 背景图片规定绘制区域

7.12 遮罩

CSS 遮罩提供一种基于像素级别的，可以控制元素透明度的能力，类似于 png24 位或 png32 位中的 Alpha 透明通道的效果。遮罩有三个属性可以设置，分别是 mask-image、mask-position、mask-repeat。

首先我们需要一张合适的遮罩图片。在 Photoshop 中操作很简单，制作步骤如下：

- (1) 打开你想要作为遮罩的透明 png24 的图片。
- (2) 选择图层菜单→图层样式→颜色叠加。
- (3) 在“颜色叠加”对话框中将颜色值改为白色。
- (4) 单击 ok 按钮关闭对话框。
- (5) 选择文件菜单，保存为 web，替换旧的图片。

执行上面的 5 个步骤，我们使用 Photoshop 制成了一个五边形，如图 7.74 所示。

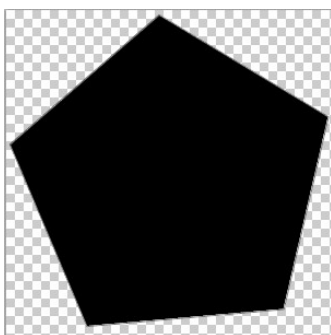


图 7.74 五边形

下例是一个简单遮罩的例子，为一个 div 设置一张背景图，再增加图 7.74 的遮罩，代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>CSS3遮罩</title>
6 <style>
7     .box{
8         width:400px;height:400px; background:url("../img/itxdl2.jpg") no-repeat;
9         background-size:cover; -webkit-mask:url("../img/mask.png") no-repeat;
10        -webkit-mask-position:50% 50%;
11    }
12 </style>
13 </head>
14 <body>
15 <div class="box"></div>
16 </body>
17 </html>
```



该 div 的遮罩图片为“mask.png”且不重复，遮罩图片的位置在“50% 50%”，在浏览器中的执行结果如图 7.75 所示。



图 7.75 CSS3 遮罩



7.13 transition 过渡

CSS3 的 transition 允许 CSS 的属性值在一定的时间区间内平滑过渡。这种效果可以在鼠标单击、获得焦点、被点击或对元素进行任何改变时触发，并圆滑地以动画效果改变 CSS 的属性值。

transition 属性是一个简写属性，用于设置四个过渡属性，这四个过渡属性的描述如表 7.21 所示。

表 7.21 transition 过渡属性及描述

属 性 名	取 值
transition-property	规定设置过渡效果的 CSS 属性的名称（all [attr] none）
transition-duration	规定完成过渡效果需要多少秒或毫秒
transition-timing-function	规定速度效果的速度曲线。 ease（逐渐变慢）默认值、linear（匀速）、ease-in（加速）、ease-out（减速）、ease-in-out（先加速后减速）、cubic-bezier 贝塞尔曲线（x1, y1, x2, y2）
transition-delay	定义过渡效果何时开始

下面，我们为一个 div 设置初始宽度为 200px，背景颜色为“red”色并增加“transition: 2s”。当鼠标移入这个 div 后，div 经过 2s 后宽度增加到 400px，背景颜色变成“pink”色。2s 的过渡时间，我们可以见证 div 的整体变化，代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>CSS3 过渡</title>
6   <style>
7     #box{ width: 200px; height: 100px; background-color: red; transition: 2s; }
8     #box:hover{ width: 400px; background-color: pink; }
9   </style>
10 </head>
11 <body>
12   <div id="box"></div>
13 </body>
14 </html>
```



将这段代码运行到浏览器中，我们可以一览 div 的过渡效果。图 7.76 所示是笔者为 div 过渡初期、中期和后期随机截取的三张图，以供读者参考。



图 7.76 div 过渡初期、中期和后期随机截图

另外，`transition` 还可以指定需要过渡的属性，如我们只需要过渡上例 `div` 的宽度属性时，只需要将 `transition` 属性的值设为 “`2s width`” 即可。这时，当我们将鼠标移入这个元素时，`div` 背景颜色立马变为 “`pink`” 色，而宽度变化是有过渡的。

我们还可以设置 `transition` 的运动样式、延迟时间及运动形式，读者感兴趣的话可以一一尝试设置运行。

7.14

2D变换

通过 CSS3 转换，能够对元素进行移动、缩放、转动、拉长或拉伸。它如何工作？转换是使元素改变形状、尺寸和位置的一种效果。CSS3 转换包括 2D 转换和 3D 转换，这里我们来了解 2D 变换的转换方法。

转换属性包含 `transform` 和 `transform-origin`，它们的介绍如表 7.22 所示。

表 7.22 转换属性介绍

属 性 名	取 值
<code>transform</code>	向元素应用 2D 或 3D 转换
<code>transform-origin</code>	允许改变被转换元素的位置



其中，transform 有 5 种方法，其方法介绍如表 7.23 所示。

表 7.23 transform 方法介绍

属 性 名	取 值
matrix(n,n,n,n,n,n)	定义 2D 转换，使用 6 个值的矩阵
translate(x,y)	定义 2D 转换，沿着 X 和 Y 轴移动元素
translateX(n)	定义 2D 转换，沿着 X 轴移动元素
translateY(n)	定义 2D 转换，沿着 Y 轴移动元素
scale(x,y)	定义 2D 缩放转换，改变元素的宽度和高度
scaleX(n)	定义 2D 缩放转换，改变元素的宽度
scaleY(n)	定义 2D 缩放转换，改变元素的高度
rotate(angle)	定义 2D 旋转，在参数中规定角度
skew(x-angle,y-angle)	定义 2D 倾斜转换，沿着 X 和 Y 轴
skewX(angle)	定义 2D 倾斜转换，沿着 X 轴
skewY(angle)	定义 2D 倾斜转换，沿着 Y 轴

7.14.1 translate()方法

通过 translate()位移函数，元素从其当前位置移动，根据给定的 left（X 坐标）和 top（Y 坐标）位置参数。translate()方法可以拆分为 translateX()和 translateY()方法，分别对元素的 left 和 top 位置设置参数。下面通过一个实例来解释这个方法，我们将 img 元素的 left 设置为 0，top 设置为 0。当鼠标移入浏览器后，让它相对于当前位置向下移动 50px，向右移动 100px，再给它设定一个过渡效果，以便读者可以更好地体会到 translate()方法实现的效果。代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>translate() 方法</title>
6 <style>
7     img{ position: relative; left: 100px; top: 100px; transition: 2s;}
8     body:hover img{ -webkit-transform:translate(100px,50px);}
9 </style>
10 </head>
11 <body>
12     
13 </body>
14 </html>
```

该代码的解释为, img 图像的 left 为 100px, 在浏览器中的执行初始效果如图 7.77 所示。

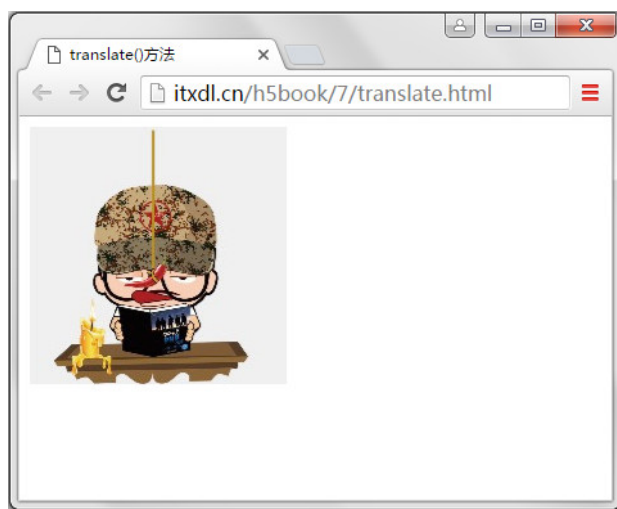


图 7.77 translate()方法 (鼠标移入前)

当鼠标移入浏览器后, 我们可以发现图片的位置发生了改变, 相对于之前的位置向下移动了 50px, 向右移动了 100px, 在浏览器中的执行效果如图 7.78 所示。

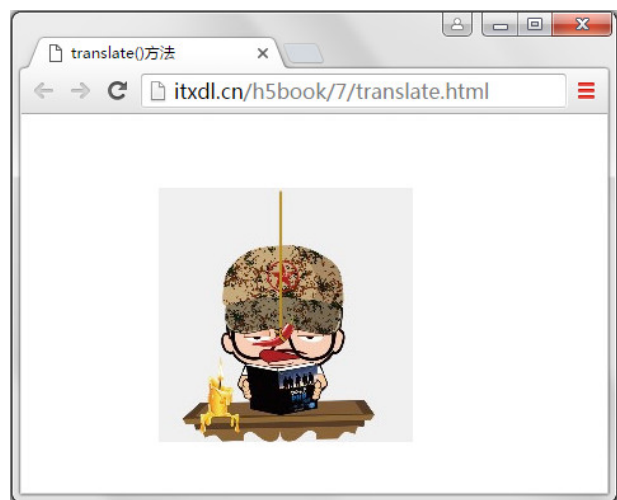


图 7.78 translate()方法 (鼠标移入后)

7.14.2 rotate()方法

通过 rotate()方法, 元素顺时针旋转给定的角度。允许负值, 元素将逆时针旋转。默认旋转基点为元素中心。



同样，我们通过一个小案例来解释这个方法。一个 HTML 文档中包含一个 `img` 元素，当鼠标移入该文档时，让图片旋转 `180deg`。代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>rotate旋转</title>
6 <style>
7     <style>
8         img{ position: relative; transition:2s; left: 100px; top: 50px;}
9         body:hover img{ transform:rotate(180deg);}
10    </style>
11 </head>
12 <body>
13     
14 </body>
15 </html>
```

该代码的解释为，`img` 图像初始状态没有旋转过，当鼠标移入 `body` 时，图片经过 2s 顺时针旋转 `180deg`，默认的旋转基点为图片中心，在浏览器中的执行初始效果如图 7.79 所示。

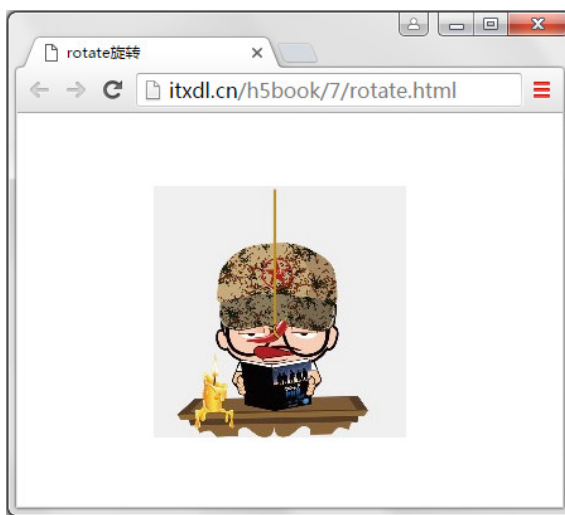


图 7.79 rotate 旋转（鼠标移入前）

当鼠标移入浏览器后，我们可以发现图片变为倒立的，也就是旋转了 `180deg`，在浏览器中的执行效果如图 7.80 所示。

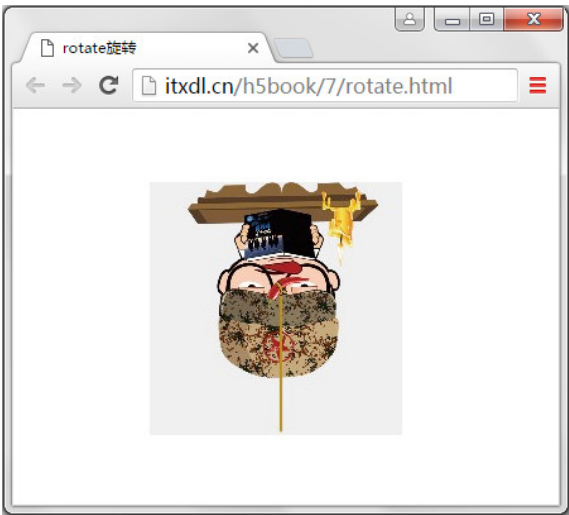


图 7.80 rotate 旋转（鼠标移入后）

另外，我们还可以为该图片设置旋转基点，transform-origin 属性允许我们改变被转换元素的位置，2D 转换元素能够改变元素 X 轴和 Y 轴。该属性的语法如下：

```
transform-origin: x-axis y-axis z-axis;
```

该属性默认值为“50% 50% 0”。针对 2D 转换，我们可以先忽略 z-axis，它是针对 3D 转换需要设定的值，我们会在下一小节提到。该属性的取值及描述如表 7.24 所示。

表 7.24 transform-origin 属性的取值及描述

值	描 述
x-axis	定义视图被置于 X 轴的何处。可能的值：left、center、right、length、%
y-axis	定义视图被置于 Y 轴的何处，可能的值：left、center、right、length、%
z-axis	定义视图被置于 Z 轴的何处，可能的值：length

通过使用 transform-origin 属性来设置转换元素的位置为左上角。将上述案例的 CSS 代码替换如下：

```
6 <style>
7   img{
8       position: relative; transition:2s; left: 200px; top: 100px;
9       /* 设置图片转换的位置为左上角 */
10      -webkit-transform-origin: left center;
11  }
12  body:hover img{ -webkit-transform:rotate(180deg); }
13 </style>
```

当鼠标移入 body 后，我们可以发现图片绕着左边界的中心点旋转了 180deg，在浏览器中的执行效果如图 7.81 所示。

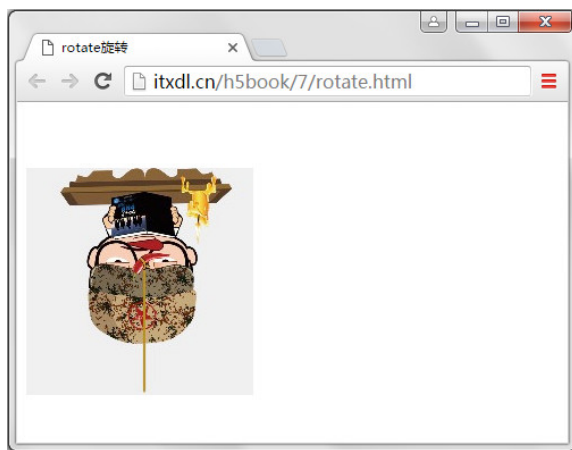


图 7.81 设置转换元素的位置

7.14.3 scale()方法

通过 `scale()` 方法，元素的尺寸会增加或减少，根据给定的宽度（ X 轴）和高度（ Y 轴）参数进行缩放。`scale()` 缩放函数让元素根据中心原点对对象进行缩放。默认值是 1，因此 0.01~0.99 之间的任何值，都使一个元素缩小；而任何大于或等于 1.01 的值，使元素显得更大。`scale()` 和 `translate()` 缩放函数的语法非常相似，它可以接受一个值，也可以同时接受两个值。当只有一个值时，其第二个值默认与第一个值相等。例如，`scale(1,1)` 元素不会有任何变化，而 `scale(2,2)` 让元素沿 X 轴和 Y 轴放大两倍。其语法如下：

```
scale(sx)
```

或者

```
scale(sx,sy)
```

该函数属性的取值及描述如表 7.25 所示。

表 7.25 `scale()` 函数属性的取值及描述

值	描述
sx	用来指定横向坐标（ X 轴）方向的缩放向量
sy	用来指定纵向坐标（ Y 轴）方向的缩放向量

同样，我们通过一个小案例来解释 `scale()` 方法。这里有两张图片并排显示，其中我们对第二张图片使用 `scale()` 放大 1.2 倍，可以使用 `scale(1.2)`，也可以使用 `scale(1.2,1.2)`，表示 X 轴和 Y 轴都放大 1.2 倍。代码如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>scale() 缩放函数</title>
6 <style>
7   img{ margin-top: 50px; border: 1px solid #000; }
8   /* 第二张图相对于原图放大1.2倍 */
9   img:nth-of-type(2){ transform:scale(1.2); }
10 </head>
11 <body>
12   
13   
14 </body>
15 </html>

```



这里，我们展示了两张图片，第一张以原图显示，第二张被放大为原图的 1.2 倍，在浏览器中的执行效果如图 7.82 所示。

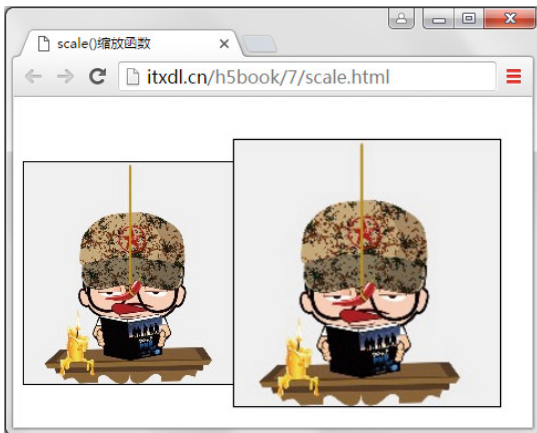


图 7.82 scale()缩放函数的执行效果

如果将值设置为“0”，则元素将消失。当我们仅给 scale()函数设置一个值时，会使对象成正比例放大或缩小。

在 scale()函数中，除了可以取正值，还可以取负值。只不过取负值时，会使元素进行翻转，然后再进行缩放。scale()函数在对元素进行缩放时，都是以元素的中心为基点，但可以通过 transform-origin 来改变元素的基点。

7.14.4 skew()方法

通过 skew()方法，元素转动给定的角度，根据给定的水平线（X 轴）和垂直线（Y 轴）参数。skew()倾斜函数能够让元素倾斜显示。它可以将一个对象以其中心位置围绕着 X 轴和 Y 轴按照一定的角度倾斜。这与 rotate()函数的旋转不同，rotate()函数只是旋转，而不会改变元素的形状。skew()函数不会旋转，而只会改变元素的形状。其语法如下：



```
skew(ax)
```

或者

```
skew(ax,ay)
```

该函数属性的取值及描述如表 7.26 所示。

表 7.26 skew()函数属性的取值及描述

值	描 述
ax	用来指定水平方向（X 轴方向）倾斜的角度
ay	用来指定垂直方向（Y 轴方向）倾斜的角度

下面我们使用一个简单的例子来说明 skew()函数，代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>skew倾斜函数</title>
6   <style>
7     img{ margin-top: 50px; border: 1px solid #000; }
8     /* 第二张图片水平方向倾斜30deg，垂直方向倾斜10deg */
9     img:nth-of-type(2){ -webkit-transform:skew(30deg, 10deg); }
10  </style>
11 </head>
12 <body>
13   
14   
15 </body>
16 </html>
```



这里，我们展示了两张图片，第一张以原图显示，第二张被 skew()函数处理后，横向倾斜 30deg，垂直倾斜 10deg。在浏览器中的执行效果如图 7.83 所示。

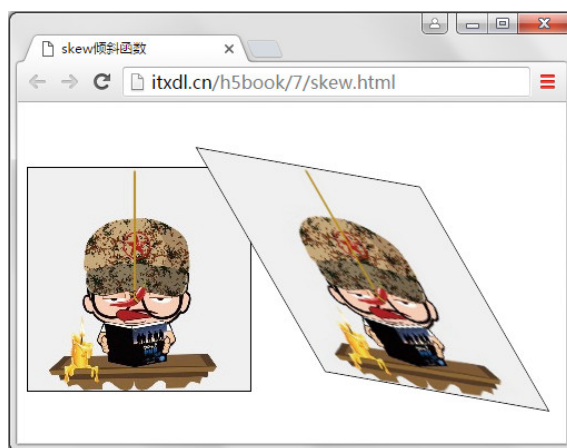


图 7.83 skew()倾斜函数的执行效果

skew() 倾斜函数和 CSS3 中变形中的 translate()、scale() 函数一样，除了可以使用 skew(ax,ay) 函数，还可以使用 skewX() 和 skewY() 函数让元素只在水平或者垂直方向倾斜。

skewX()：相当于 skew(ax,0) 和 skew(ax)。按给定的角度沿 X 轴指定一个倾斜变形。skewX() 使元素以其中心为基点，并在水平方向（X 轴）进行倾斜变形。

skewY()：相当于 skew(0,ay)。按给定的角度沿 Y 轴指定一个倾斜变形。skewY() 使元素以其中心为基点，并在垂直方向（Y 轴）进行倾斜变形。

在默认情况下，skew() 函数都是以元素的原中心点对元素进行倾斜变形的。但是我们同样可以根据 transform-origin 属性，重新设置元素基点对元素进行倾斜变形。

7.15

3D变换

3D 变换较 2D 变换多了下转换属性，3D 转换属性及描述如表 7.27 所示。

表 7.27 3D 转换属性及描述

属 性	描 述
transform	向元素应用 2D 或 3D 转换
transform-style	规定被嵌套元素如何在 3D 空间显示
perspective	规定 3D 元素的透视效果、景深
perspective-origin	规定元素的底部位置
back-visibility	定义元素在不面对屏幕时是否可见

3D 的转换方法如表 7.28 所示。

表 7.28 3D 的转换方法

属 性	描 述
matrix(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	定义 3D 转换，使用 16 个值得 4×4 矩阵
translate3d(x,y,z)	定义 3D 转化
translateX(x)	定义 3D 转化，仅使用 X 轴的值
translateY(y)	定义 3D 转化，仅使用 Y 轴的值
translateZ(z)	定义 3D 转化，仅使用 Z 轴的值
scale3d(x,y,z)	定义 3D 缩放转换



续表

属 性	描 述
scaleX(x)	定义 3D 缩放转换，通过给定一个 X 轴的值
scaleY(y)	定义 3D 缩放转换，通过给定一个 Y 轴的值
scaleZ(z)	定义 3D 缩放转换，通过给定一个 Z 轴的值
rotate3d(x,y,z,angle)	定义 3D 旋转
rotateX(angle)	定义沿 X 轴的 3D 旋转
rotateY(angle)	定义沿 Y 轴的 3D 旋转
rotateZ(angle)	定义沿 Z 轴的 3D 旋转
perspective(n)	定义 3D 转换元素的透视视图

7.15.1 transform-style

`transform-style` 经常被用来做三维空间坐标系中的图形变换，下面我们就来看看 CSS3 中设置 3D 变形的 `transform-style` 属性详解。`transform-style` 指定嵌套元素如何在 3D 空间中呈现，主要有两个属性：`flat` 和 `perserve-3d`。

`transform-style` 属性的使用语法如下：

```
transform-style: flat | perserve-3d;
```

其中 `flat` 值为默认值，表示所有子元素在 2D 平面呈现；`perserve-3d` 表示所有子元素在 3D 空间中呈现。`transform-style` 属性需要设置在父元素中，并且高于任何嵌套的变形元素。

7.15.2 perspective景深

`perspective` 景深属性对于 3D 变形来说至关重要。该属性会设置查看者的位置，并将可视内容映射到视锥上，继而投到一个 2D 视平面上。如果不指定透视，则 Z 轴空间中的所有点将平铺到同一个 2D 平面上，并且变换结果中将不存在景深的概念。其实对于 `perspective` 属性，我们可以简单地理解为视距，用来设置用户和元素 3D 空间 Z 平面之间的距离。而其效应由它决定，值越小，用户和 3D 空间 Z 平面距离越近，视觉效果越深刻；反之，值越大，用户与 3D 空间 Z 平面距离越远，视觉效果越不深刻。

`perspective` 属性的使用语法如下：

```
perspective: none | <length>;
```

`perspective` 属性包括两个属性：`none` 和具有单位的长度值`<length>`。其中 `perspective` 属性的默认值为 `none`，表示无限的角度来看 3D 物体，但看上去是平面的。另一个属性`<length>`接受一个长度单位大于 0 的值，而且其单位不能为百分比值。`<length>`值越大，角度出现得越远，从而创建一个低强度的角度和非常小的 3D 空间变化；反之，此值越小，角度出现得越近，从而创建一个高强度的角度和一个大型的 3D 空间变化。

下面通过一个小案例来加深一下对 `perspective` 的印象。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>perspective 景深</title>
6   <style>
7     .wrap{
8       width: 200px; height: 200px; border: 1px solid red; float: left; margin-right: 30px;
9       /* 建立3D空间 */
10      transform-style: preserve-3d;
11    }
12    .wrap img{
13      /* 图片沿X轴方向旋转45deg */
14      transform: rotateY(45deg);
15    }
16    .wrap:nth-of-type(2){
17      /* 设置景深为600px */
18      perspective: 600px;
19    }
20  </style>
21 </head>
22 <body>
23   <div class="wrap">
24     
25   </div>
26   <div class="wrap">
27     
28   </div>
29 </body>
30 </html>
```



这里有两个 `div`，每个 `div` 中包含一张图片。每个 `div` 都建立 3D 空间，让图片绕着 Y 轴旋转 45deg。不同的是，我们为第二个 `div` 设置 600px 的景深，第一张不设置。在浏览器中查看，运行效果如图 7.84 所示。

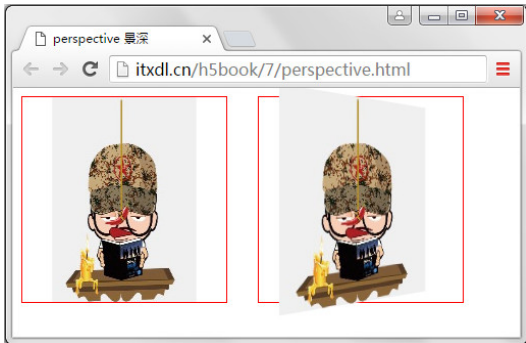


图 7.84 perspective 景深的运行效果



从图 7.84 中我们可以看出两张图的差别。在父节点没有设置 `perspective` 的情况下，图片的 3D 旋转效果不明显，而在父节点设置 `perspective` 后，图片才有 3D 旋转的效果。

7.15.3 perspective-origin 景深基点

`perspective-origin` 景深基点属性是 3D 变形中另一个重要属性，主要用来决定 `perspective` 属性的源点角度。它实际上设置了 `X` 轴和 `Y` 轴位置，在该位置观看者好像在观看该元素的子元素。

`perspective-origin` 属性的使用语法如下：

```
perspective-origin: tx ty;
```

该属性的默认值为“50% 50%”，可以设置一个值，也可以设置两个长度值。其取值及描述如表 7.29 所示。

表 7.29 perspective-origin 属性的取值及描述

值	描 述
tx	指定相对于元素的包含框的 <code>X</code> 轴上的位置，可能的取值：<percentage> <length> left center right
ty	指定相对于元素的包含框的 <code>Y</code> 轴上的位置，可能的取值：<percentage> <length> top center bottom

同 `perspective` 一样，`perspective-origin` 属性必须定义在父元素上。通常 `perspective-origin` 属性本身不做任何事情，它必须被定义在设置了 `perspective` 属性的元素上。换句话说，`perspective-origin` 属性需要与 `perspective` 属性结合使用，以便将视点移至元素的中心以外的位置。

同样，我们通过一个小案例来加深一下对 `perspective-origin` 的印象。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>perspective-origin 景深基点</title>
6   <style>
7     .wrap{
8       width: 200px; height: 200px; border: 1px solid red; float: left; margin-right: 30px;
9       /* 建立3D空间 */
10      transform-style: preserve-3d;
11      /* 设置景深为600px */
12      perspective: 600px;
13    }
14    .wrap img{
15      /* 图片沿X轴方向旋转45deg */
16      transform: rotateY(45deg);
17    }
18    .wrap:nth-of-type(2){
19      /* 设置景深基点为left top */
20      perspective-origin: left top;
21    }
```

```

22     </style>
23 </head>
24 <body>
25     <div class="wrap">
26         
27     </div>
28     <div class="wrap">
29         
30     </div>
31 </body>
32 </html>

```



在这里我们也设置了两个 div，每个 div 中包含一张图片。每个 div 都建立 3D 空间，设置 600px 的景深。不同的是，我们将第二个 div 的景深基点改为“left top”，第一个 div 使用默认的景深基点“50% 50%”。在浏览器中进行查看，运行效果如图 7.85 所示。

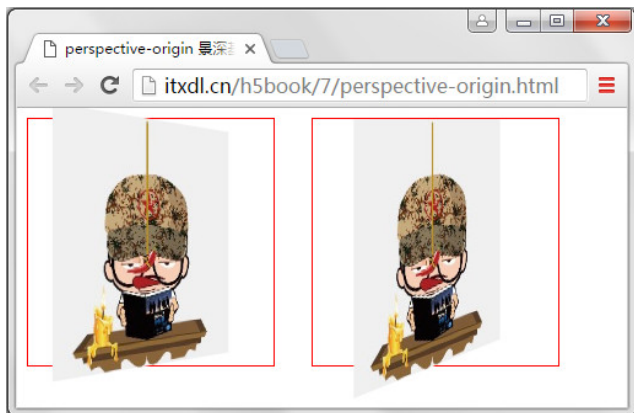


图 7.85 perspective-origin 景深基点的运行效果

从图 7.85 中我们可以看出两张图的差别。设置景深基点相当于换了位置查看这个 3D 变化。

7.15.4 3D位移

在 CSS3 中 3D 位移主要包括两种函数：`translateZ()`和 `translate3d()`。`translate3d()`函数使一个元素在三维空间移动。这种变形的特点是，使用三维向量的坐标定义元素在每个方向移动了多少。其基本语法如下：

```
translate3d(tx,ty,tz)
```

其属性取值及说明如表 7.30 所示。



表 7.30 translate3d(tx,ty,tz)属性值取值及说明

值	描 述
tx	代表横向坐标位移向量的长度
ty	代表纵向坐标位移向量的长度
tz	代表 Z 轴位移向量的长度。此值不能是一个百分比值，如果取值为百分比值，则会认为是无效值

下面来看一个简单的实例，加深对 translate3d() 的理解。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>3D位移</title>
6   <style>
7     .wrap{
8       width: 200px; height: 200px; border: 1px solid red; float: left; margin-right: 30px;
9       /* 建立3D空间 */
10      transform-style: preserve-3d;
11      /* 设置景深为600px */
12      perspective: 600px;
13    }
14    .wrap:nth-of-type(2) img{
15      /* 图片位移移动，横向坐标为10px、纵向坐标为10px、Z轴位移偏移量为-100px */
16      transform: translate3d(10px, 10px, -100px);
17    }
18  </style>
19 </head>
20 <body>
21   <div class="wrap">
22     
23   </div>
24   <div class="wrap">
25     
26   </div>
27 </body>
28 </html>
```



这里我们对第二张图片使用 translate3d() 方法，让它相对于之前的位置偏移，横向偏移 10px，纵向偏移 10px，Z 轴位移偏移量为 -100px。在浏览器中进行查看，运行效果如图 7.86 所示。

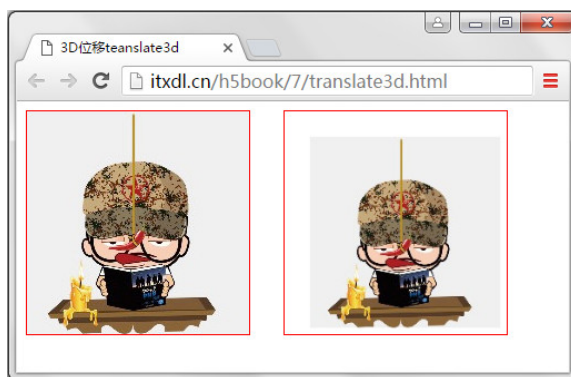


图 7.86 3D 位移 (translate3d()) 效果图

从图 7.86 中我们可以看出两张图的差别。第二张图位移发生了改变，向右偏移了 10px，向下偏移了 10px，向后偏移了 100px。

在 CSS3 中，3D 位移除了 `translate3d()` 函数，还有 `translateZ()` 函数。`translateZ()` 函数的功能是让元素在 3D 空间沿 Z 轴进行位移，其基本使用语法如下：

`translateZ(t)`

t 指的是 Z 轴的向量位移长度。

使用 `translateZ()` 函数可以让元素在 Z 轴进行偏移，当其为负值时，元素在 Z 轴越移越远，导致元素变得较小；反之，当其值为正值时，元素在 Z 轴越来越远，导致元素变得较大。将上例 CSS 代码的 `translate3d()` 方法替换为 `translateZ()` 方法，代码如下：

```

7  .wrap{
8      width: 200px; height: 200px; border: 1px solid red; float: left; margin-right: 30px;
9      /* 建立3D空间 */
10     transform-style: preserve-3d;
11     /* 设置景深为600px */
12     perspective: 600px;
13 }
14 .wrap:nth-of-type(2) img{
15     /* 图片Z轴位移偏移量为-100px */
16     transform: translateZ(-100px);
17 }

```



这里我们对第二张图片使用 `translateZ()` 方法，Z 轴位移偏移量为 -100px。在浏览器中进行查看，运行效果如图 7.87 所示。

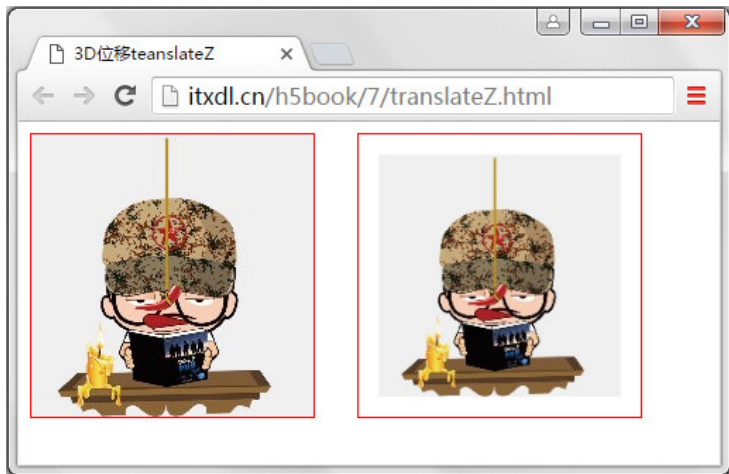


图 7.87 3D 位移 (`translateZ()`) 效果图

`translateZ()` 函数仅让元素在 Z 轴进行位移，在实际使用中等同于 `translate3d(0,0,tz)`。



7.15.5 3D旋转

在三维变形中，我们可以让元素在任何轴旋转。为此，CSS3 新增了三个旋转函数：`rotateX()`、`rotateY()`和 `rotateZ()`。使用 `rotateX()`函数允许一个元素围绕 *X* 轴旋转；使用 `rotateY()`函数允许一个元素围绕 *Y* 轴旋转；使用 `rotateZ()`函数允许一个元素围绕 *Z* 轴旋转。

`rotateX()`函数指定一个元素围绕 *X* 轴旋转，旋转的量被定义为指定的角度。如果值为正值，则元素围绕 *X* 轴顺时针旋转；反之，如果值为负值，则元素围绕 *X* 轴逆时针旋转。其基本语法如下：

```
rotateX(a)
```

其中，*a* 指的是一个旋转角度值，其值可以是正值，也可以是负值。

`rotateY()`函数指定一个元素围绕 *Y* 轴旋转，旋转的量被定义为指定的角度。如果值为正值，则元素围绕 *Y* 轴顺时针旋转；反之，如果值为负值，则元素围绕 *Y* 轴逆时针旋转。其基本语法如下：

```
rotateY(a)
```

其中，*a* 指的是一个旋转角度值，其值可以是正值，也可以是负值。

`rotateZ()`函数和其他两个函数功能一样，区别在于 `rotateZ()`函数指定一个元素围绕 *Z* 轴旋转。其基本语法如下：

```
rotateZ(a)
```

`rotateZ()`函数指定元素围绕 *Z* 轴旋转，如果仅从视觉角度上看，`rotateZ()`函数让元素顺时针或逆时针旋转，并且效果和 `rotate()`效果等同，但它不是在 2D 平面的旋转。

下面来看一个简单的实例，加深对 `rotate()`函数的理解。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>3D旋转</title>
6   <style>
7     .wrap{
8       width: 200px; height: 200px; border: 1px solid red; float: left; margin: 40px 10px;
9       /* 建立3D空间 */
10      transform-style: preserve-3d;
11      /* 设置景深为600px */
12      perspective: 600px;
13    }
14    .wrap:nth-of-type(1) img{
15      /* 图片绕X轴旋转30deg */
16      transform: rotateX(30deg);
17    }
18    .wrap:nth-of-type(2) img{
19      /* 图片绕Y轴旋转30deg */
20      transform: rotateY(30deg);
21    }
```

```

22     .wrap:nth-of-type(3) img{
23         /* 图片绕Z轴旋转30deg */
24         transform: rotateZ(30deg);
25     }
26 }
27 </style>
28 </head>
29 <body>
30     <div class="wrap">
31         
32     </div>
33     <div class="wrap">
34         
35     </div>
36     <div class="wrap">
37         
38     </div>
39 </body>
</html>

```



这里我们让第一张图片绕 X 轴旋转 30deg，第二张图片绕 Y 轴旋转 30deg，第三张图片绕 Z 轴旋转 30deg。在浏览器中查看，运行效果如图 7.88 所示。

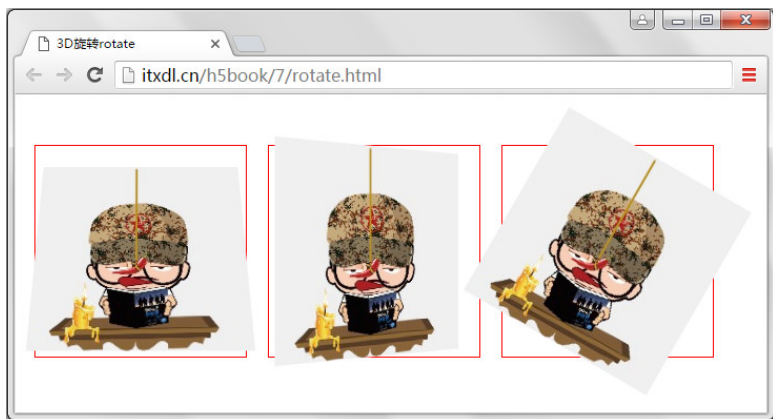


图 7.88 3D 旋转效果图

7.15.6 3D 缩放

在 CSS3 中，3D 变形中的缩放主要有 `scaleZ()` 和 `scale3d()` 两种函数。当 `scale3d()` 中的 X 轴和 Y 轴同时为 1 时，即 `scale3d(1,1,sz)`，其效果等同于 `scaleZ(sz)`。通过使用 3D 缩放函数，可以让元素在 Z 轴上按比例缩放。默认值为 1，当值大于 1 时，元素放大；反之，小于 1 大于 0.01 时，元素缩小。其使用语法如下：

```
scale3d(sx,sy,sz)
```

`scale3d()` 属性取值及描述如表 7.31 所示。



表 7.31 scale3d()属性取值及描述

值	描述
sx	横向缩放比例
sy	纵向缩放比例
sz	Z 轴缩放比例

2D 的缩放函数为 `scale(sx,sy)`，能横向和纵向缩放元素。3D 转化多了一个 Z 轴的缩放。用 `scaleZ()` 函数实现元素在 Z 轴的缩放，其使用语法如下：

`scaleZ(s)`

s 指定元素每个点在 Z 轴的比例。下面一起来看一个简单的实例，加深对 `scaleZ()` 函数的理解。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>3D缩放scaleZ()</title>
6   <style>
7     .wrap{
8       width: 200px; height: 200px; border: 1px solid red; float: left; margin: 40px 10px;
9       /* 建立3D空间 */
10      transform-style: preserve-3d;
11      /* 设置景深为600px */
12      perspective: 600px;
13    }
14    .wrap img{
15      /* 图片绕Y轴旋转30deg */
16      transform: rotateY(30deg);
17    }
18    .wrap:nth-of-type(2) img{
19      /* 图片绕Y轴旋转30deg, 放大3倍 */
20      transform: scaleZ(3) rotateY(30deg);
21    }
22  </style>
23 </head>
24 <body>
25   <div class="wrap">
26     
27   </div>
28   <div class="wrap">
29     
30   </div>
31 </body>
32 </html>
```



这里我们让第一张图片绕 Y 轴旋转 30deg，第二张图片绕 Y 轴旋转 30deg 并放大 3 倍。在浏览器中进行查看，运行效果如图 7.89 所示。

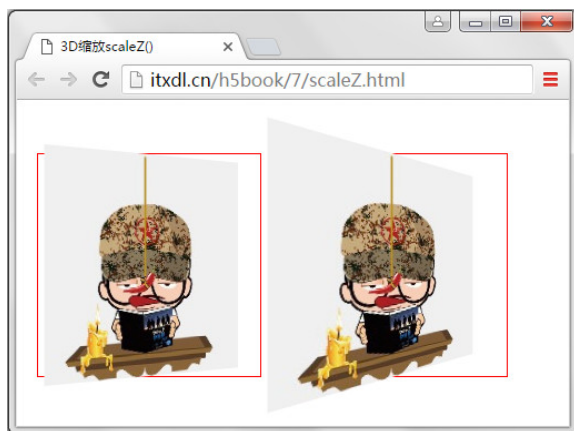


图 7.89 3D 缩放函数 scaleZ()

7.15.7 3D盒子

这里使用 3D 转换的属性及方法构造出一个 6 面的 3D 盒子，该盒子是一个正方体。代码如下：

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <title>3D盒子</title>
6   <style>
7     .wrap{
8       width:100px;height:100px;padding:10px; border:1px solid #000; margin:100px auto;
9       /* 该div是个2D转换，景深为200px，景深基点为中心，X轴和Y轴放大2倍 */
10      perspective:200px; perspective-origin:center center; transform:scale(2);
11    }
12    .box{
13      width:100px;height:100px;background:red; position:relative; transition:3s;
14      /* 该div是3D盒子，先申明一个3D空间，景深X轴、Y轴和Z轴的中心，也就是正方体的中心 */
15      transform-style:preserve-3d; transform-origin:center center -50px;
16    }
17    .box div{
18      width:100px;height:100px; position:absolute; color:#fff;font-size:50px;
19      text-align:center;line-height:100px;
20    }
21    /* 下面将每个面布置到指定的位置 */
22    .box div:nth-of-type(1){
23      left:0;top:-100px;background:#9C0; transform-origin:bottom; transform:rotateX(90deg);
24    }
25    .box div:nth-of-type(2){
26      left:-100px;top:0px;background:#CF3; transform-origin:right; transform:rotateY(-90deg);
27    }
28    .box div:nth-of-type(3){
29      left:0px;top:0px;background:#CCF;
30    }
31    .box div:nth-of-type(4){
32      left:100px;top:0;background:#0C9; transform-origin:left;transform:rotateY(90deg);
33    }
34    .box div:nth-of-type(5){
35      left:0px;top:100px;background:#69C; transform-origin:top;transform:rotateX(-90deg);
36    }
37    .box div:nth-of-type(6){
38      left:0;top:0;background:#F0C; transform:translateZ(-100px) rotateX(180deg);

```



```
39 }
40 /* 当鼠标移入到3D盒子内时, 让该盒子绕X轴旋转一圈 */
41 .wrap:hover .box{ transform:rotateX(360deg);}
42 </style>
43 </head>
44 <body>
45 <div class="wrap">
46   <div class="box">
47     <div>1</div>
48     <div>2</div>
49     <div>3</div>
50     <div>4</div>
51     <div>5</div>
52     <div>6</div>
53   </div>
54 </div>
55 </body>
56 </html>
```

这样就构造出了一个 3D 盒子, 正面是 3。当鼠标移入盒子时, 让该盒子绕 X 轴旋转 360deg, 为了让读者清晰地看到盒子旋转的过程, 我们为该盒子增加了过渡。下面是旋转过程的随机截图, 如图 7.90 所示。

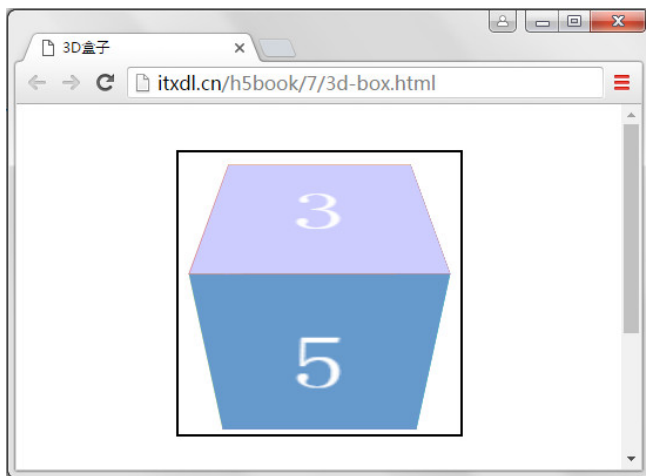


图 7.90 3D 盒子

7.16 animation 动画

CSS3 属性中关于制作动画的三个属性为 transform、transition 和 animation。前面已经介绍过 transform 和 transition, 这里我们来学习 animation 动画属性。通过 CSS3, 我们能够创建动画, 可以在许多网页中取代动画图片、Flash 动画及 JavaScript。

7.16.1 关键帧keyframes

如需在 CSS3 中创建动画，我们需要先学习@keyframes 规则。前面所提到的 transition 制作的过渡效果，包括了初始属性和最终属性，一个开始执行动作时间和一个延续动作时间以及动作的变换速率，其实这些值都是中间值。如果我们想要控制得更细一些，比如我们要第一个时间执行什么动作，第二时间执行什么动作（就如 Flash 中说的第一帧要执行什么动作，第二帧要执行什么动作），那么用 transition 就很难实现了，此时我们就需要这样的一个“关键帧”来控制，而 CSS3 的 animation 的“keyframes”属性可以实现这样的效果。

@keyframes 规则用于创建动画，在@keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。@keyframes 的格式如下：

```
@keyframes 动画名称{
    from{
        properties: properties value;
    }
    percentage{
        properties: properties value;
    }
    to{
        properties: properties value;
    }
}
```

或者

```
@keyframes 动画名称{
    0%{
        properties: properties value;
    }
    percentage{
        properties: properties value;
    }
    100%{
        properties: properties value;
    }
}
```

其中 percentage 是百分比值，我们可以添加许多个这样的百分比；properties 为 CSS 的属性名，比如 left、background 等；value 就是相对应的属性的属性值。值得一提的是，from 和 to 分别对应的是 0% 和 100%。这个我们在前面也提到了。

下面一起来看一个简单的实例，代码如下：

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5     <title>animation动画</title>
6     <style>
```



```
7  /* 申明一个名叫"itxd1"的关键帧 */
8  @keyframes itxd1{
9      form{
10         background: red;
11         width: 100px;
12     }
13     50%{
14         background: blue;
15         width: 200px;
16     }
17     to{
18         background: green;
19         width: 400px;
20     }
21 }
22 .wrap{
23     /* 调用"itxd1"的动画属性, 过渡时间为3s */
24     width:100px; height:100px; border:1px solid #000; background:red; animation:3s itxd1;
25 }
26 </style>
27 </head>
28 <body>
29     <div class="wrap">
30     </div>
31 </body>
32 </html>
```



CSS animation 动画效果将会影响元素相对应的 CSS 值, 在整个动画过程中, 元素的变化属性值完全由 animation 来控制, 动画后面的属性值会覆盖前面的属性值。如上面的例子, 因为这个案例只是在不同的时间段改变了 div 的背景色和宽度, 其默认值是: width:100px; background:red; 但我们在执行动画“from”时, div 属性发生改变: width:100px;background:red; 当执行到 50%时, 属性变成了: width:200px;background:blue; 当动画执行到“from”时, 属性变为: width:400px;background: green; 此时动画将完成, 那么 width 和 background 两个属性值将以“to”时的为主, 不会产生叠加效果, 只是一次一次覆盖前一次出现的 CSS 属性值。就如我们平时的 CSS 一样, 最后出现的权限是最大的。当动画结束后, 样式回到默认效果。下面是该动画过程的截图, 如图 7.91 所示。

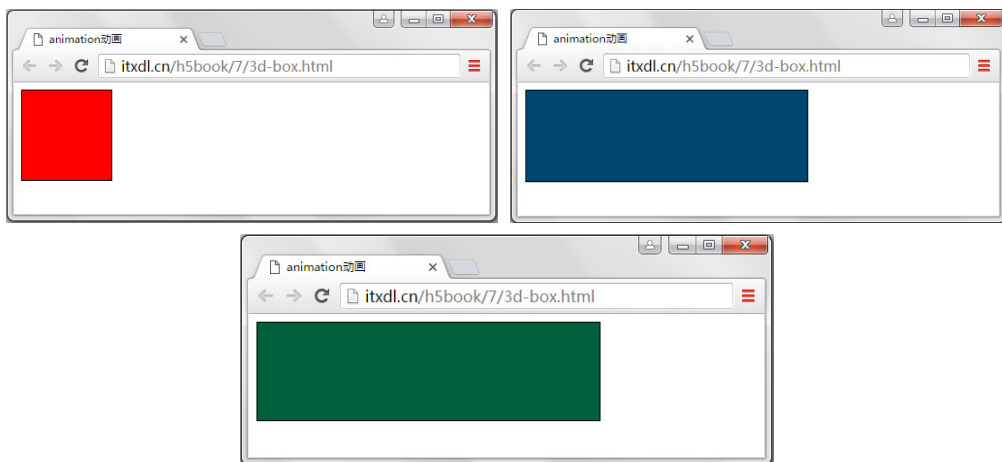


图 7.91 animation 动画过程截图

7.16.2 animation动画属性

animation 的动画属性及描述如表 7.32 所示。

表 7.32 animation 的动画属性及描述

值	描 述
animation	所有动画属性的简写属性，除了 animation-play-state 属性
animation-name	规定@keyframes 动画的名称
animation-duration	规定动画完成一个周期所花费的时间，取值为数值，单位为 s（秒）。默认是 0
animation-timing-function	规定动画的速度曲线。可能的取值：ease（逐渐变慢）、linear（匀速）、ease-in（加速）、ease-out（减速）、ease-in-out（加速然后减速）、cubic-bezier（自定义）。默认是 ease
animation-delay	规定动画何时开始。取值为数值，单位为 s（秒）。默认是 0
animation-iteration-count	规定动画被播放的次数。取值为数值，默认是 1。Infinite 为无限次数循环
animation-direction	规定动画是否在下一周期逆向播放，可能的取值有 normal、alternate。默认是 normal
animation-play-state	规定动画是否正在运行或暂停。可能的取值有 running、paused。默认是 running
animation-fill-mode	规定对象动画时间之外的状态

animation 的动画属性包含必要属性和可选属性。必要属性包括 animation-name（动画名称）、animation-duration（动画持续时间）、animation-play-state（播放状态），其余属性为可选属性。

对于表 7.32 中的属性，我们可以分别设置。这里以 animation-timing-function 规定动画的速度曲线为例。修改动画的速度曲线为“ease-in-out”，使动画先加速然后减速。修改 wrap 的 CSS 代码如下：

```
22 .wrap{
23     /* 调用"itxd1"的动画属性，过渡时间为3s */
24     width:100px; height:100px; border:1px solid #000; background:red; animation:3s itxd1;
25     /* 将动画的速度曲线修改为ease-in-out，使动画先加速后减速 */
26     animation-timing-function: ease-in-out;
27 }
```

其余的动画属性也是这样设置，读者感兴趣的话，可以为每个动画属性设置不同的取值来深入了解 animation。

本章小结

CSS3 新增了许多属性，CSS3 样式新增了一种颜色模式 RGBA，用来制作透明色，比



CSS 的颜色模式多了一个透明度的设置。文字的 CSS3 特效有文字阴影、文字描边、文字排版和文字省略等。另外，CSS3 增加了弹性盒模型，我们可以为盒模型添加阴影和倒影。CSS3 的分栏布局让 Web 开发人员轻松实现让文本呈现多列显示。传统的圆角生成方案，必须使用多张图片作为背景图案。CSS3 的出现，使得我们再也不必浪费时间去制作这些图片了，仅需要通过设置 `border-radius` 属性即可。通过 CSS3，我们能够创建圆角边框，向矩形添加阴影，使用图片来绘制边框。CSS3 `gradient` 分为 `linear-gradient`（线性渐变）和 `radial-gradient`（径向渐变）。CSS3 规范中对背景这一部分，新加入了设定多个背景图片、指定背景大小、设置背景渐变等功能。遮罩提供一种基于像素级别的，可以控制元素透明度的能力，类似于 `png24` 位或 `png32` 位中的 Alpha 透明通道的效果。`transition` 允许 CSS 的属性值在一定的时间区间内平滑过渡。2D 变换和 3D 变换能够对元素进行移动、缩放、转动、拉长或拉伸。制作动画需要使用 `animation` 属性，执行动作需要由关键帧 `@keyframes` 来控制。

本章习题

1. 在 CSS 中,注释代码是 ()。
A. `<!-- 注释代码 >` B. `/* 注释代码 */`
C. `// 注释代码` D. `|| 注释代码`
2. 要实现溢出文本显示省略号效果, 需要增加怎样的样式? ()
A. `text-overflow:ellipsis;`
B. `text-overflow:ellipsis; white-space:nowrap;`
C. `text-overflow:ellipsis; white-space:nowrap; overflow:hidden;`
D. `white-space:nowrap; overflow:hidden;`
3. 以下哪段代码不能将宽、高都为 200px 的 div 设置为圆角? ()
A. `div{ border: 1px solid #000; border-radius: 100px; }`
B. `div{ border: 1px solid #000; border-radius: 100px 100px; }`
C. `div{ border: 1px solid #000; border-radius: 50%; }`
D. `div{ border: 1px solid #000; border-radius: 100%; }`
4. 下面哪段代码不能实现颜色从顶部向底部渐变? ()
A. `background-image: -webkit-linear-gradient(white,black);`
B. `background-image: -webkit-linear-gradient(top,white,black);`
C. `background-image: -webkit-linear-gradient(90deg,white,black);`
D. `background-image: -webkit-linear-gradient(-90deg,white,black);`
5. 下列属性哪一个能够实现层的隐藏? ()

第8章

DIV+CSS 网页标准化布局



标准的网页需要对内容进行布局，以前都采用表格的定位技术，从 2005 年开始逐步转向 DIV+CSS 的布局方式，目前绝大多数网站都采用这种布局方式。使用 DIV+CSS 对网站进行布局符合 W3C 标准，采用这种方式布局通常是为了说明与 HTML 表格定位方式的区别。通过使用 DIV 盒子模型结构将各部分内容划分到不同的区块，然后用 CSS 来定义盒子模型的位置、大小、边框、内外边距、排列方式等。简单地说，DIV 用于搭建网站结构（框架），CSS 用于创建网站表现（样式/美化）。该标准简化了 HTML 页面代码，获得了一个较优秀的网站结构，有利于日后网站维护、协同工作和便于搜索引擎抓取。当然，并不是所有的网页都需要用 DIV 布局，如数据页面、报表等页面，还是使用 HTML 表格比较方便，Web 标准中并没有说要抛弃表格。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

8.1 DIV+CSS 页面布局的优势

DIV+CSS 是一种网页的布局方式，在 HTML 网站设计标准中，不再使用表格定位技术，而是采用 DIV+CSS 的方式实现各种定位。DIV+CSS 是网站标准（也称“Web 标准”）中的常用术语之一，可实现网页页面内容与表现相分离。采用 DIV+CSS 对网站进行构建越来越

被网页设计者重视,因为在 DIV+CSS 结构的网页中,几乎每个元素的属性都是使用 CSS 定义的,所以对网页区域和元素的调整起到了很大的作用。这种 DIV+CSS 模式的网站设计具有以下优势。

1. 表现和内容相分离

在 HTML 文件中只存放文本信息,而将设计部分放在一个独立样式文件中,使我们能够对页面的布局施加更多的控制,HTML 代码仍然可以保持简单明了的初衷。

2. 代码简洁,提高页面浏览速度

CSS 的极大优势表现为简洁的代码。对于一个大型网站来说,不仅可以节省大量的带宽、提高页面的浏览速度,而且增加了有效关键词占网页总代码的比重。因此使用 DIV+CSS 的 Web 标准制作的网站,对搜索引擎抓取页面具有一定的优势。

3. 易于维护和改版

网页内容和设计样式的分离,减少了一些重复构建的方法,使页面和样式的调整变得更加方便,只要简单地修改几个 CSS 文件,就可以重新设计整个网站的页面。很多问题只需要改变 CSS 而不需要改动程序,从而降低了网站改版的成本。

4. 提高搜索引擎对网页的索引效率

使用 DIV+CSS 方式构建的网站容易向 W3C 标准靠拢,因为网站是否符合 W3C 标准是搜索引擎对网页排名的一个影响因素。另外,网站源代码简洁,除了<div>、、、等标签,几乎不用其他标签。而且使用 DIV+CSS 可以把网页中的主要内容放在前面,这样网站的内容就完全展现在搜索引擎面前,便于搜索引擎抓取关键内容。

8.2 “无意义”的HTML标签<div>和

HTML 只是赋予内容的手段,大部分 HTML 标签都有其意义(例如,标签<a>创建链接,标签<h1>创建标题等),然而<div>和标签却似乎没有任何内容上的意义,听起来就像一把泡沫做成的锤子一样无用。但实际上,与 CSS 结合后,它们的应用就变得十分广泛。需要记住的是,<div>和是“无意义”的标签,它们的存在纯粹是为了应用 CSS 样式,所以当样式表失效时,它们就不再起任何作用。

<div>和被用来组合成一大块的 HTML 代码并赋予一定的信息,大部分用类属性 class 和标识属性 id 与元素联系起来。和<div>的不同之处在于,是内联的(行内标记),用在一小块的内联 HTML 中;而<div>(division)是块级的(简单地说,它等同于其前后有断行),用于组合一大块的代码,为 HTML 文档内大块的内容提供结构和背景的元素,可以包含段落、标题、表格,甚至是其他部分,这使得<div>便于建立不同集成的类。



```
1 <div id="scissors">                                <!-- 使用div组合一大块的代码 -->
2   <p>This is <span class="paper">crazy</span></p>    <!-- 使用span内联在p标记中 -->
3 </div>                                              <!-- 使用div结束一个块 -->
```

<div>的起始标签和结束标签之间的所有内容都是用来构成这个块的，其中所包含的元素特性由<div>标签的属性来控制，或者通过使用样式表格式化这个块来进行控制。

8.3 W3C盒子模型

日常生活中所见的盒子也就是一种能装东西的箱子，如果家里的东西很多，则需要按类别装到不同的箱子中。网页中的内容表现也是一样的，如果页面内容比较多，又想让页面更整洁、更美观、有很好的用户体验，那么也需要按类别划分到不同的区块中，划分出来的每个区块就可以看作一个装东西的盒子。而每个 HTML 元素都可以看作一个区块，类似于装了东西的盒子，所以称其为盒子模型。在盒子模型中，除了可以装内容（文字、图片等），还可以再装小盒子，所以一个页面的布局就是使用多个盒子按顺序摆放或嵌套组合成页面框架，再在不同的盒子中放入对应的网页内容。

如何去定义一个盒子呢？又如何去摆放盒子布局页面呢？通常我们使用“无意义”的标签<div>来定义一个盒子模型，再通过 CSS 属性去声明盒子模型的属性。一个盒子的属性包括它的宽度（width）和高度（height），盒子里面的内容到盒子的边框之间的距离即填充（padding），盒子本身有边框（border），而盒子边框外和其他盒子之间还有边界（margin），如图 8.1 所示。

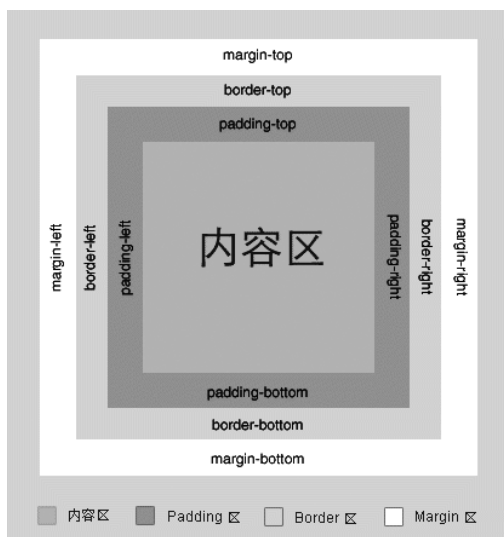


图 8.1 W3C 盒子模型

该盒子模型的声明代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8" />
5   <title>盒子模型</title>
6   <style>
7     #box{                                /* id为box的盒子模型          */
8       width: 200px;                      /* 盒子的宽度为200px          */
9       height: 200px;                     /* 盒子的高度为200px         */
10      border: 5px solid #ccc;             /* 盒子边框实线各边宽5px    */
11      padding: 10px;                      /* 盒子的4个内填充为10px    */
12      margin: 20px;                       /* 盒子的4个外边距为20px    */
13    }
14  </style>
15 </head>
16 <body>
17   <div id="box">
18     内容区
19   </div>
20 </body>
21 </html>

```



用浏览器打开包含这段 HTML 代码的文件，效果如图 8.2 所示。



图 8.2 W3C 盒子模型效果图

在使用 CSS 对网页进行布局时，盒子模型是页面布局的基础。上例中则是由 id 为 #box 的 <div> 元素声明的矩形框，这个框由元素内容、空白、填充及边框等组成。而在 CSS 中使用 width 和 height 属性指定区块的宽度和高度分别为 200px，使用 border 属性添加 4 个边宽度都为 5px 的边框线，通过 margin 和 padding 属性分别定义区块外边距为 20px 和内填充为 10px，如图 8.2 所示。

使用 margin 指定外边距是为了设置与其他区块之间的距离，而使用内填充的目的是在



内容与边框之间创建一个隔离带，使内容不会与边框混在一起。定义盒子模型所需要的 CSS 属性如表 8.1 所示。

表 8.1 声明盒子模型的 CSS 属性

属 性	描 述
margin	定义区块外边界与上级元素距离的属性，用 1~4 个值来设置元素的边界，每个值都是长度、百分比或 auto。百分比值参考上级元素的宽度，允许使用负值。如果 4 个值都给出了，则它们分别被应用于上、右、下和左边界。如果只给出一个值，则它被应用于所有边界。如果给出了两个或三个值，则省略了的值与对边相等。注意，如果边界在垂直方向邻接（重叠）了，则会改用其中最大的那个边界值，而水平方向则不会。也可以选择使用上边界 margin-top、下边界 margin-bottom、左边界 margin-left 和右边界 margin-right 属性分别设置与上级元素的外边距
padding	用于设置区块的内边距属性，是边框和元素内容之间的间隔距离。与 margin 属性相反，但使用的是相同的属性值，是上补白 padding-top、右补白 padding-right、下补白 padding-bottom 和左补白 padding-left 属性的略写
border	边框属性，用于设置一个元素的边框风格、边框宽度、边框颜色，可以一起设置 4 条边的边框，也可对上边框、右边框、下边框和左边框进行单独设置
width	盒子的宽度，可以用一个长度或“auto”值来指定其宽度，不允许使用负值
height	盒子的高度，可以用一个长度或“auto”值来指定其高度，不允许使用负值

盒子区块外边距和内填充距离都是可选的，大部分元素的默认值为 0，但有一些元素会存在非 0 的默认值，如 body、ul、h3 等，会给页面布局带来不便。可以将元素的 margin 和 padding 属性设置为 0，清除这些默认的空白。在网页设计中，内容常指文字、图片等元素，但也可以是小盒子（DIV 嵌套）。与现实生活中的盒子不同的是，在现实生活中东西一般不能大于盒子，否则盒子会被撑坏；而 CSS 盒子具有弹性，盒子中的东西大过盒子本身最多把它撑大，而不会被损坏。填充只有宽度属性，可以理解为生活中盒子里的抗震辅料厚度。而边框有大小和颜色之分，我们又可以理解为生活中所见盒子的厚度，以及这个盒子是用什么颜色的材料做成的。边界就是该盒子与其他东西要保持多大距离。

8.4 和页面布局有关的 CSS 属性

使用 DIV+CSS 对网页进行标准化布局前，除了要掌握盒子模型，还要掌握“定位”和“浮动”两个比较重要的概念，它们可以控制在页面上排列和显示元素的方式。一个盒子是装内容的区块，如果多个盒子组合在一起使用，再通过定位和浮动进行设置，就可以对整个页面进行布局。图 8.3 所示为由多个盒子布局的页面，每个虚线框代表一个盒子模型。

虽然 CSS 的样式属性非常多，但实际参与页面布局的属性却很少。CSS 的定位属性应

用得非常广泛，可以控制元素的平面或空间位置，以及高度、宽度和可见性。也可以使用 CSS 的 `display` 属性改变生成区块的类型，例如将 `display` 属性设置为 `none`，则这个区块框及其所有内容就不再显示。通过将 `display` 属性设置为 `block`，可以让行内元素表现得像块级元素一样。常见的参与页面布局的 CSS 属性如表 8.2 所示。



图 8.3 多个盒子定义页面布局

表 8.2 常见的参与页面布局的 CSS 属性

属 性	描 述
<code>position</code>	用于定义一个元素是否 <code>absolute</code> （绝对）、 <code>relative</code> （相对）、 <code>static</code> （静态）或者 <code>fixed</code> （固定）
<code>top</code>	层距离页面顶点纵坐标的距离
<code>left</code>	层距离页面顶点横坐标的距离
<code>text-align</code>	横向排列，可以使用 <code>left</code> （居左对齐）、 <code>right</code> （居右对齐）、 <code>center</code> （居中对齐）值
<code>line-height</code>	指定行高，内容都在行的中间，所以可以使用这个属性设置内容垂直居中。这个属性接受一个控制文本基线之间的间隔的值，当值为数字时，行高由元素字体大小的量与该数字相乘所得。百分比的值相对于元素字体的大小而定。不允许使用负值
<code>z-index</code>	决定层的先后顺序和覆盖关系，值高的元素会覆盖值比较低的元素
<code>display</code>	<code>display</code> 是一个显示属性，设定为 <code>block</code> 值则以块状显示，在元素后面添加换行符，即其他元素不能在其后面并列显示。如果设定为 <code>inline</code> 值则内联显示，在元素后面删除换行符，多个元素可以在一行内并列显示。使用 <code>none</code> 值将关闭指定元素及其子元素的显示



续表

属 性	描 述
visibility	这个属性是针对嵌套层的设置，如果存在嵌套的层（子层）和被嵌套的层（父层），则可以使用 inherit 值设置子层继承父层的可见性。如果父层可见，则子层也可见。当使用 visible 值时，无论父层是否可见，子层都可见。而当值为 hidden 时，无论父层是否可见，子层都隐藏
overflow	用于设置层内的内容超出层所能容纳的范围的处理方式。为该属性设置 visible 值时，无论层的大小如何，内容都会显示出来。当设置 hidden 值时，会隐藏超出层大小的内容。当设置值为 scroll 时，不管内容是否超出层的范围，选中此项都会为层添加滚动条。而当使用 auto 值时，只在内容超出层的范围时才显示滚动条
float	设置区块漂浮属性，允许网页制作者将文本环绕在一个元素的周围，可以使用左漂浮 left 值和右漂浮 right 值
clear	清除属性，指定一个元素是否允许有元素漂浮在它的旁边。值 left 移动元素到其左边漂浮的元素下面；值 right 移动元素到其右边漂浮的元素下面。其他的还有默认的 none 值，以及移动元素到其两边漂浮的元素下面的 both 值

在 CSS 中提供了相对和绝对两种定位方法。所谓相对定位，是指让操作的元素在相对其他元素的位置上进行偏移；而绝对定位，是指让操作的元素参照原始文档进行偏移。使用表 8.2 中部分定位属性的例句代码如下：

```
1 #box {                               /* 声明 id 选择器，名称为 box */           */
2     position: absolute;               /* 设置层的定位为绝对定位 */           */
3     top: 30px;                       /* 层距离顶点纵坐标的距离为 30px */      */
4     left: 100px;                     /* 层距离顶点横坐标的距离为 100px */     */
5     width: 300px;                    /* 设置层的宽度为 300px */               */
6     height: 150px;                  /* 设置层的高度为 150px */              */
7     overflow: auto;                  /* 当内容超出层的范围时显示滚动条 */    */
8     z-index: 1;                     /* 设置层的先后顺序为覆盖关系 */        */
9     visibility: visible;             /* 无论父层是否可见，子层都可见 */      */
10 }
```

8.5 盒子区块框的定位

区块的定位有普通流、绝对定位和浮动 3 种基本的定位机制。如果不是专门指定区块的位置，则默认是在普通流中定位，即从上到下一个接一个地排列，位置由元素在 HTML 中的位置决定。如果使用像 span 和 strong 等不自动换行的行内元素，就会在一行中水平布局。可以通过使用水平填充、外边距等调整它们的水平间距。

8.5.1 相对定位

相对定位通常会被看作普通流定位的一部分，因为元素的位置相对于它本身的普通流中

的位置定位并不是布局的常用方式。如果某个区块框在它所在的位置处，设置垂直或水平位置，就可以让这个区块“相对于”它在普通流的起点位置进行移动。但在使用相对定位时，无论是否移动，元素仍然会占据原来的空间，因此，这种移动方式会导致它覆盖其他的区块。在下面的例子中实现将鼠标光标移动到页面的链接上时，链接的元素就会在网页中颤动一下。

```
1 a:hover {                                /* 定义a元素的伪选择器，当鼠标光标移动到链接上时改变样式 */
2     position:relative;                   /* 设置元素使用相对定位 */
3     top:1px;                             /* 鼠标进入时a元素将出现在原位置顶部下面1px的地方 */
4     left:1px;                           /* 鼠标进入时a元素将出现在原位置右边1px的地方 */
5 }
```

在本例中，当鼠标光标放在超链接上时，链接元素就会相对于原位置向下移动 1px 并向右移动 1px。

8.5.2 绝对定位

相对定位是相对于自身在普通流中的位置移动，而绝对定位使元素的位置与文档的普通流无关，它的位置相对于已定位的包含它的上层元素上、下、左、右移动。如果没有已定位的上层元素，那么它的位置相对于最初的包含区块移动，如 body 或 HTML 元素。

可以直接将元素定位在页面上的任何位置，而且绝对定位不会占据普通流中现有区块框的空间，其他元素在布局时可以把绝对定位的元素视为不存在，这样就提供了非常灵活的布局方式。也可以完全通过绝对定位的方式布局整个页面，但每个区块都需要使用 width 和 height 属性设置固定的尺寸；否则如果扩大绝对定位中的某个区块，则它周围使用绝对定位的区块不会重新定位。因此，尺寸的任何改变都将会导致绝对定位的区块产生重叠，从而破坏精心调整过的页面布局。绝对定位的简单应用代码如下：

```
1 <style>
2     #demo {                               /* 定义一个id选择器 */
3         position:absolute;               /* 使用绝对位置进行定位 */
4         width:300px;                     /* 定义盒子宽度为300px */
5         height:300px;                    /* 定义盒子高度为300px */
6         top:100px;                       /* 定义盒子距离网页顶部100px */
7         left:200px;                      /* 定义盒子距离网页左边200px */
8         background:#BABABA;              /* 定义盒子的背景颜色为灰色 */
9         z-index:1;                       /* 定义盒子位于上一层中 */
10    }
11 </style>
12
13 <div id="demo">我是一个盒子区块，我现在在网页中的哪个位置呢？</div>
```

网页中漂浮的区块及在页面中浮动的广告，都必须采用绝对定位的机制。因为绝对定位的区块与普通文档流无关，所以它们可以覆盖在页面中其他区块的上面。也可以通过设置 z-index 属性来控制这些区块的堆放次序，z-index 的值越大，区块在层中的位置就越高。



使用绝对定位去布局页面的情况比较少见，大多数绝对定位都是为了配合 JavaScript 去完成一些页面特效，当然也有一些页面布局需要使用绝对定位来完成。例如，大多数网站在进入管理平台之前都需要先登录，而这个登录框的盒子模型如果使用绝对定位来完成，就可以做到在调整浏览器大小时登录框的盒子一直在浏览器中居中。例如，登录框的盒子模型代码设计如下：

```
1 <html>
2   <head>
3     <title>登录框的盒子模型定位</title>
4     <style>
5       #login {                                /* 定义一个id选择器 */
6         width:300px;                          /* 定义盒子宽度为300px */
7         height:200px;                        /* 定义盒子高度为200px */
8         position:absolute;                  /* 使用绝对位置进行定位 */
9         left:50%;                            /* 左部盒子开始位置是页面宽度的50% */
10        top:50%;                             /* 顶部盒子开始位置是页面高度的50% */
11        margin-left:-150px;                  /* 左部开始位置再退回盒子宽度的一半 */
12        margin-top:-100px;                  /* 顶部开始位置再退回盒子高度的一半 */
13        background:#BABABA;                 /* 定义盒子的背景颜色为灰色 */
14      }
15    </style>
16  </head>
17  <body>
18    <div id="login">
19      登录框的盒子模型
20    </div>
21  </body>
22 </html>
```

8.6 使用盒子模型的浮动布局

虽然使用绝对定位可以实现页面布局，但由于调整某个盒子模型时其他盒子模型的位置并不会跟着改变，所以并不是布局的首选方式。而使用浮动的盒子模型可以向左或向右移动，直到它的外边缘碰到包含它的盒子模型边框或另一个浮动盒子模型的边框为止。并且由于浮动的盒子模型不在文档的普通流中，所以文档的普通流中的盒子模型表现得就像浮动的盒子模型不存在一样。

8.6.1 设置浮动

在 CSS 中，我们通过 float 属性实现盒子区块框向左或向右浮动。其实任何元素都可以浮动，而浮动元素会生成一个块级框，不论它本身是何种元素。但浮动的元素要指定一个明确的宽度，否则它们会尽可能地窄。

在图 8.4 所示的两张图片中，左图是按普通的文档流布局的三个盒子区块框，它们从上

到下一个接一个地排列，位置由元素在 HTML 中的位置决定。而在右图中，当把第一个区块框向右浮动时，它脱离普通文档流并且向右移动，直到它的右边缘碰到包含框的右边缘，而其他两个区块框就会在普通流中上移。

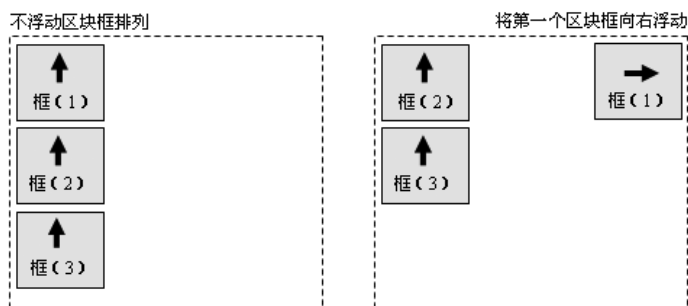


图 8.4 CSS 浮动属性的应用

另外，在图 8.4 所示的两张图片中，当将第一个区块框设置向左浮动时，它就会脱离文档流并且向左移动，直到它的左边缘碰到包含框的左边缘。这样，第一个区块框就不再处于文档流中，所以它也不占据空间，则第二个区块框就会在文档流中自动上移，但被设置左浮动的第一个区块框覆盖住了，从而使第二个区块框从视图中消失（见图 8.5 左图）。对于这种情况，可以对布局中的所有东西进行浮动。如果把三个区块框都向左移动，那么第一个区块框向左浮动直到碰到包含框，另外两个区块框向左浮动直到碰到前一个区块框，呈现在一行中排列，如图 8.5 右图所示。

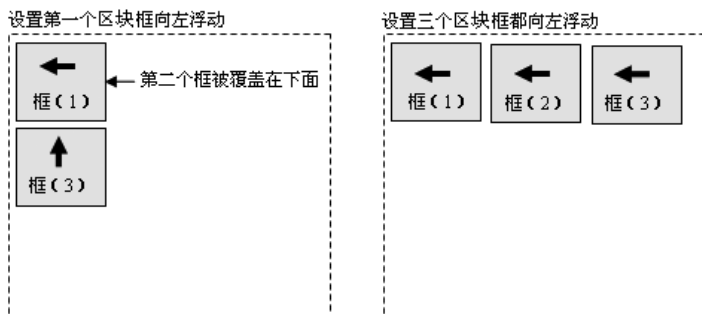


图 8.5 浮动设置的对比演示

假如在一行之上只有极少的空间可供区块框浮动，那么这个区块框会跳至下一行，这个过程会持续到某一行拥有足够的空间为止，如图 8.6 左图所示。而如果浮动元素的高度不同，那么当它们向下移动时，可能会被其他浮动元素“卡住”，如图 8.6 右图所示。

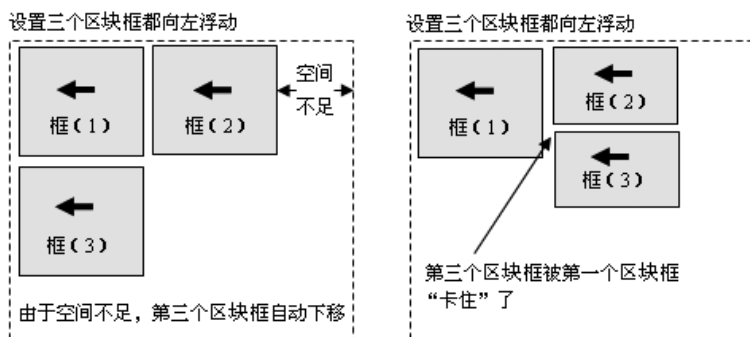


图 8.6 空间不足的区块框浮动演示

8.6.2 行框和清理

在进行页面布局时，经常需要设置多个区块框并列在一行中排列。最常见的方式就是使用 `float` 属性，再通过 `left` 或 `right` 值移动区块框向左或向右浮动。但当前面并列的多个区块框总宽度不足以包含框的 100% 时，就会在行框中留出一定的宽度，而下面的某个区块框又恰好满足这个宽度，则很可能会向上提，和上一行并列的区块框在同一行排列。而这并不是我们想要的结果，所以可以使用 `clear` 属性解决这一问题。该属性的值可以是 `left`、`right`、`both` 或 `none`，它表示框的哪些边不应该挨着浮动框。代码如下：

```
1 <html>
2   <head>
3     <style>
4       .left {                                /* 声明一个CSS类选择器，名字为left */
5         width:200px;                          /* 设置盒子模型的宽度为200px */
6         height:200px;                        /* 设置盒子模型的高度也为200px */
7         margin:10px;                         /* 设置盒子模型的外边距为10px */
8         border:solid 1px;                    /* 设置盒子有1px的实线边框 */
9         float:left;                          /* 设置盒子向左浮动，脱离了文档流 */
10      }
11     .noleft {                                /* 声明另一个CSS类选择器，名字为noleft */
12       width:200px;                          /* 设置盒子模型的宽度为200px */
13       height:200px;                        /* 设置盒子模型的高度为200px */
14       border:solid 1px;                    /* 设置盒子有1px的实线边框 */
15       background:#ccc;                     /* 设置盒子模型背景为灰色 */
16     }
17   </style>
18 </head>
19 <body>
20   <div class="left"> 框（一）</div>          <!--使用类left,设置左浮动,脱离了文档流 -->
21   <div class="left"> 框（二）</div>          <!-- 也使用类left,也设置左浮动,和第一个盒子在同一行 -->
22   <div class="noleft"> 框（三）</div>        <!-- 使用类noleft,没有设置浮动,在文档流中 -->
23 </body>
24 </html>
```

如果不清除浮动，那么第三个区块框就会和第一、第二个区块框显示在一行中，又因为第一、第二个区块框设置了浮动就脱离了文档流，所以第三个区块框会在第一个区块框的下

面，如图 8.7 所示。如果我们在第三个区块框的样式中加一个清除浮动 `clear:both`，则设置第三个区块框两边都不能挨着浮动框，所以第三个区块框会在下一行独立出现，如图 8.8 所示。

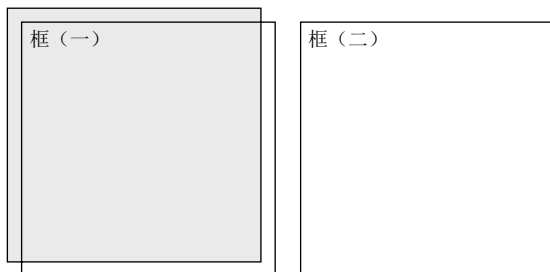


图 8.7 没有设置清除浮动

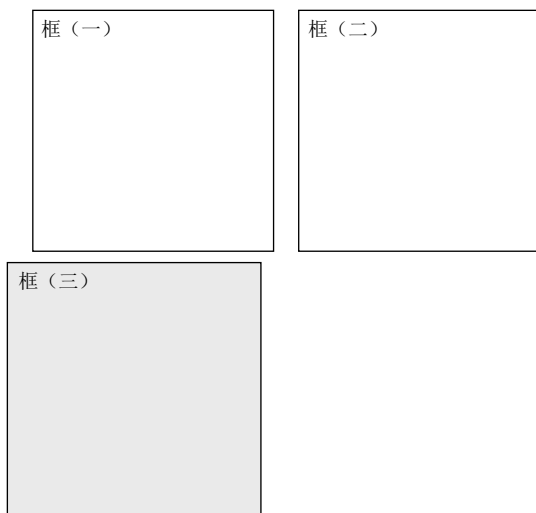


图 8.8 设置清除浮动

对于以上这种情况，我们使用最多的方法是将“清除浮动”单独定义一个 CSS 样式，然后使用单独一个区块框来专门进行“清除浮动”。代码如下：

```

1 <html>
2   <head>
3     <style>
4       .left {                               /* 声明一个CSS类选择器，名字为left */
5         width:200px;                         /* 设置盒子模型的宽度为200px */
6         height:200px;                       /* 设置盒子模型的高度也为200px */
7         margin:10px;                       /* 设置盒子模型的外边距为10px */
8         border:solid 1px;                   /* 设置盒子有1px的实线边框 */
9         float:left;                         /* 设置盒子向左浮动，脱离了文档流 */
10      }
11     .noleft {                               /* 声明另一个CSS类选择器，名字为noleft */
12       width:200px;                         /* 设置盒子模型的宽度为200px */
13       height:200px;                       /* 设置盒子模型的高度为200px */
14       border:solid 1px;                   /* 设置盒子有1px的实线边框 */

```



```
15      background:#ccc;          /* 设置盒子模型背景为灰色          */
16    }
17    .clear {                     /* 声明一个独立的浮动清除样式类          */
18      clear:both;               /* 设置两边都不挨着浮动框          */
19    }
20  </style>
21 </head>
22 <body>
23   <div class="left"> 框（一） </div>    <!--使用类left,设置左浮动,脱离了文档流          -->
24   <div class="left"> 框（二） </div>    <!-- 也使用类left,也设置左浮动,和第一个盒子在同一行 -->
25   <div class="clear"> </div>            <!-- 这个区块专门用于清除浮动,相当于一个分隔符号 -->
26   <div class="noleft"> 框（三） </div> <!-- 使用类noleft,没有设置浮动,在文档流中          -->
27 </body>
28 </html>
```

8.7 DIV+CSS的兼容性问题



使用 DIV+CSS 布局网页其实是很容易的事情,但各种浏览器之间的不兼容性问题,加大了页面布局的难度,给程序员带来很多不便,于是在调试各种浏览器的兼容性上需要花费更多的时间。因为部分 CSS 属性在不同的浏览器之间解析的结果会有差异,这是由于个别浏览器的开发商对一些 CSS 属性的解析没有按 W3C 的标准设计而造成的。这也是初学者常常会认为布局难以理解的原因。逃避不是解决问题的办法,因为每种浏览器都会有它的使用人群,一个好的网站布局需

要在所有浏览器上都可以看到相同的界面。就算不能调试成各种浏览器显示完全一样,也要保证大致相同。

可以使用的浏览器虽然有很多种,但通常在进行页面布局调试时将其划分为 IE 和非 IE 两类。主要原因是微软公司的 IE 浏览器没有按 W3C 的标准设计,而非 IE 的浏览器则绝大部分是符合 W3C 标准的,如 Safari 和 Firefox 等浏览器对 CSS 的解析相差无几。另外,更令人烦恼的是,IE 还有多种版本 (IE 5、IE 6、IE 7、IE 8 等),对 CSS 的解析也存在很大差异。虽然针对 IE 的兼容性调试很费时,但还是要多花费一些精力在这上面,因为毕竟 IE 浏览器占有很大的市场份额。但目前 IE 5 的使用人群少到可以不用去考虑它了,一般 IE 浏览器的兼容性只针对 IE 6、IE 7 和 IE 8 即可,IE 的每个新版本的出现都在向标准化迈进,非 IE 的浏览器使用 Firefox 浏览器为代表即可。DIV+CSS 进行页面布局时需要处理的兼容性问题有很多,本节只能给出部分常见的兼容性问题的解决方案。

8.7.1 不同浏览器解释盒子模型的差异

盒子模型是 CSS 中一个重要的概念,理解了盒子模型才能更好地排版。不同浏览器对盒子模型的解释各不相同。其实盒子模型有两种,分别是 IE 盒子模型(IE 系列浏览器)和标准 W3C 盒子模型(非 IE 浏览器),如图 8.9 所示。

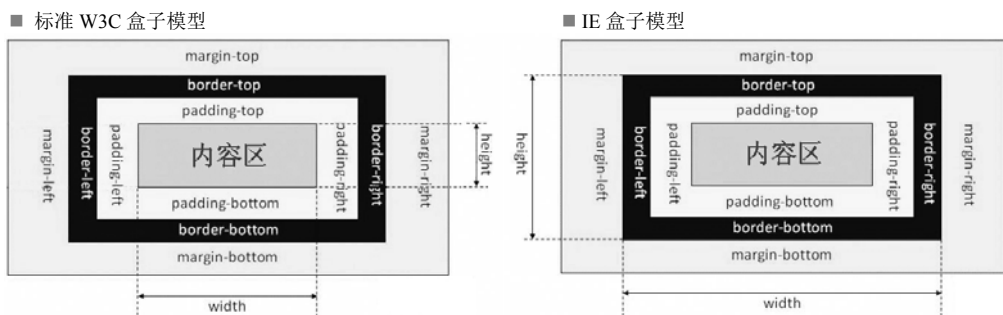


图 8.9 两种盒子模型

从图 8.9 左图可以看到,标准 W3C 盒子模型的范围包括 margin、border、padding、内容区,并且内容区部分不包含其他部分。而从图 8.9 右图可以看到,IE 盒子模型的范围也包括 margin、border、padding、内容区,和标准 W3C 盒子模型不同的是,IE 盒子模型的内容区部分包含了 border 和 padding。例如,一个盒子模型的 margin 为 20px, border 为 1px, padding 为 10px, 内容的宽为 200px、高为 50px。代码如下:

```

1 <style>
2     #box {                               /* 定义主盒子区块样式 */
3         margin: 20px;                    /* 盒子的外部边距为20px */
4         padding: 10px;                   /* 盒子的内部边距为10px */
5         border: 1px solid #000;          /* 盒子的边框线为黑色1px的直线 */
6         width: 200px;                    /* 盒子内容区块的宽度为200px */
7         height: 50px;                    /* 盒子内容区块的高度为50px */
8     }
9 </style>
10 <div id="box">
11     <!-- 使用div定义一个盒子模型作为一个区块 -->
12 </div>

```

如果用标准 W3C 盒子模型解释(见图 8.10),那么这个盒子模型需要占据的位置(包含外边距 margin)和盒子模型的实际大小(不包含外边距 margin)如下。

盒子模型占据的位置:

宽度 = margin×2 + border×2 + padding×2 + 内容的 width = 20×2 + 1×2 + 10×2 + 200 = 262px

高度 = margin×2 + border×2 + padding×2 + 内容的 height = 20×2 + 1×2 + 10×2 + 50 = 112px

盒子模型的实际大小:

宽度 = border×2 + padding×2 + 内容的 width = 1×2+10×2+200=222px

高度 = border×2 + padding×2 + 内容的 height = 1×2+10×2+50=72px



如果用 IE 盒子模型解释(见图 8.11),那么这个盒子需要占据的位置(包含外边距 margin)和盒子模型的实际大小(不包含外边距 margin)如下。

盒子模型占据的位置:

宽度 = $\text{margin} \times 2 + \text{内容的 width} = 20 \times 2 + 200 = 240\text{px}$

高度 = $\text{margin} \times 2 + \text{内容的 height} = 20 \times 2 + 50 = 90\text{px}$

盒子模型的实际大小:

宽度 = 内容的 width = 200px

高度 = 内容的 height = 50px

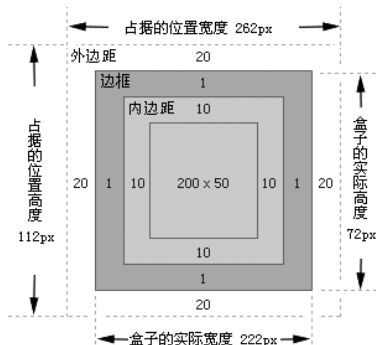


图 8.10 标准 W3C 盒子模型解释

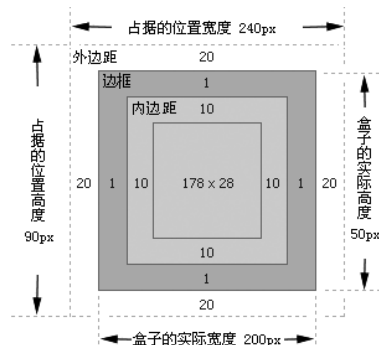


图 8.11 IE 盒子模型解释

8.7.2 遵循W3C标准设置浏览器

8.7.1 节的介绍让我们知道了不同浏览器对盒子模型的解释是有所差异的,那么应该选择哪种盒子模型呢?答案一定是“标准 W3C 盒子模型”。怎样才算是选择了“标准 W3C 盒子模型”呢?只要能让 IE 浏览器也按“标准 W3C 盒子模型”去解析页面即可。当然,不只是盒子模型可以让 IE 浏览器按标准的 W3C 规范去解析,整个页面的 CSS 都可以让 IE 浏览器按标准的 W3C 规范去解析。那么应该怎样去做呢?其实很容易,就是在网页的顶部加上 DOCTYPE 声明。如果不加 DOCTYPE 声明,那么各个浏览器会根据自己的行为去理解网页。例如,IE 浏览器会采用 IE 盒子模型解释你的盒子,而 Firefox 浏览器会采用标准 W3C 盒子模型解释你的盒子,所以网页在不同的浏览器中显示得就不一样。反之,如果加上了 DOCTYPE 声明,那么所有浏览器都会采用标准 W3C 盒子模型解释你的盒子,网页就能在各个浏览器中显示一致了。

DOCTYPE 定义当前文档的基本类型,即文档类型定义(DTD),用于告诉浏览器打开页面时应遵循什么规则。要建立符合标准的网页,DOCTYPE 声明是必不可少的组成部分。DOCTYPE 声明及声明位置如图 8.12 所示。



图 8.12 DOCTYPE 声明及声明位置

其实 DOCTYPE 只是一组机器可读的规范，虽然中间包含了文件的 URL，但浏览器不会读取这些文件，仅用于识别，然后决定以什么样的规范执行页面中的代码。开始制作符合标准的站点时，第一件事情就是声明符合自己需要的 DOCTYPE。而 XHTML 1.0 提供了 3 种 DTD 声明可供选择，分别为过渡的（Transitional）、严格的（Strict），以及专门针对框架页面设计使用的 DTD（Frameset）。这里笔者推荐 DOCTYPE 声明是过渡的 DTD，这也是最常用的 DOCTYPE 声明，声明代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 提供的这种过渡的 DTD，其要求是非常宽松的，它允许继续使用 HTML 4.01 的标识，但是要符合 XHTML 的写法。笔者推荐使用最新的、简便的 DOCTYPE 声明方法，只需几个字符即可，代码如下：

```
<!DOCTYPE html>
```

虽然声明 DOCTYPE 解决了大部分问题，但还是会有个别的标记和样式不能兼容；或是不去声明 DOCTYPE，我们就需要针对不同的浏览器去写不同的 CSS，让它能够同时兼容不同的浏览器，使得在不同的浏览器中也能得到我们想要的页面效果。

8.8 使用盒子模型设计页面布局

布局所涉及的技术非常多，足以另写一本书了。在本节中主要介绍网站中最常用的布局方案，包括区块框居中、两列浮动、三列浮动及多列浮动的布局。

8.8.1 居中设计

区块框居中设计是网页布局中常用的技术，例如将网页内的主体内容设置为一定的宽度，然后在页面内居中占据屏幕的一部分显示，而不是横跨整个屏幕宽度。这样设计是因为现在的显示器尺寸越来越大，网页的可读性问题变得越来越重要，因而需要创建比较短的、容易阅读的行。另外，不要让使用低分辨率的显示器用户，通过反复拖动浏览器的水平滚动



条来查看页面中的每行文本。目前网页的布局几乎都是基于 1024×768 像素的屏幕分辨率来设计页面内容的宽度。例如，将页面显示内容区块框的宽度设置为 966 像素，代码如下：

```
1 <html>
2   <head>
3     <title>居中设计</title>
4     <style>
5       body {
6         margin:0;           /* 为网页主体内容区域设置样式 */
7         padding:0;          /* 设定网页外部边距值为0，消除body默认值 */
8         text-align:center;  /* 设定网页内部边距值为0，消除body默认值 */
9       }                    /* 为了在IE中设置主体容器盒子居中 */
10      #container {
11        width:966px;         /* 为布局的最外层容器使用id选择器设置样式 */
12        margin:0 auto;      /* 设置最外层容器宽度为966px */
13        text-align:left;    /* 设置外边距上下为0，左右自动，则在Firefox中居中 */
14        background:#888;    /* 再将主容器中的文本内容调回为居左显示 */
15        height:800px;       /* 临时设置一下背景颜色显示主容器布局效果 */
16      }                    /* 也是临时设置一下高度显示主容器的布局效果 */
17    </style>
18  </head>
19  <body>
20    <div id="container">    <!-- 使用css消除主体标记默认的边距，设置文本居中 -->
21      最外层的容器div在屏幕上水平居中
22    </div>                <!-- 定义网页最外层的容器使用css设置居中显示 -->
23  </body>
24 </html>
```

在上面的代码中，设置页面最外层的容器 div 水平居中。在 CSS 中只需要定义容器 div 的宽度为 966 像素，然后将容器的左部和右部空白边设置为 auto，就会在浏览器中居中。但在没有声明 DOCTYPE 时，IE 浏览器中不支持使用空白边设置容器居中，而是通过在上一层容器中设置样式 text-align:center 让容器居中，然后再将容器的内容重新对准左边。如图 8.13 所示，IE 和 Firefox 浏览器显示结果相同。

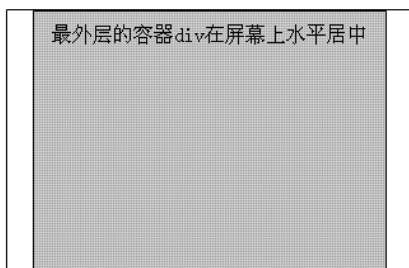


图 8.13 居中设计

8.8.2 设置两列浮动的布局

使用 float 设置的浮动盒子区块会脱离文档流，并且不会占据文档流中的任何空间，所以浮动的盒子区块就不再对包含它们的区块框产生任何影响。另外，在一行内如果有足够的

空间，则多个设置浮动的盒子就会显示在同一行中。所以使用浮动的盒子区块去创建简单的两列布局，只要同一行有足够的空间并将两个盒子都设置为浮动即可。例如，在很多网站中常见的主导航以边条的形式显示在左边、主体内容区域在右边显示的两列，我们将主导航区块框和内容区块框两列设计包含在一个居中容器 div 中，这个 div 就使用前面介绍的方法设计为水平居中。代码如下：

```

1 <html>
2   <head>
3     <title>设置两列浮动</title>
4     <style>
5       body{ margin:0; padding:0; text-align:center; }
6       #container { width:966px; margin:0 auto; text-align:left; }
7       #left_main {          /* 设置左部导航区块的css布局样式 */
8         float:left;          /* 设置该区块框向左浮动，脱离文档流 */
9         width:256px;          /* 设定该区块框的宽度为256px */
10        height:400px;         /* 设定该区块框的高度为400px,临时设置 */
11        border:1px solid;      /* 设定该区块框的边框为1px的直线边框 */
12      }
13      #right_content {
14        float:right;           /* 设置该区块框向右浮动，脱离文档流 */
15        width:700px;           /* 设定该区块框的宽度为700px */
16        height:400px;          /* 设定该区块框的高度为400px,临时设置 */
17        border:1px solid;      /* 设定该区块框的边框为1px的直线边框 */
18      }
19    </style>
20  </head>
21  <body>
22    <div id="container">
23      <div id="left_main">
24        主导航区块
25      </div>
26      <div id="right_content">
27        内容区块
28      </div>
29    </div>
30  </body>
31 </html>

```

在上面的代码中，非常简单地实现了两列浮动的布局，只为每列设置想要的宽度，然后将主导航所在的区块框向左浮动，将内容区块框向右浮动，如图 8.14 所示。

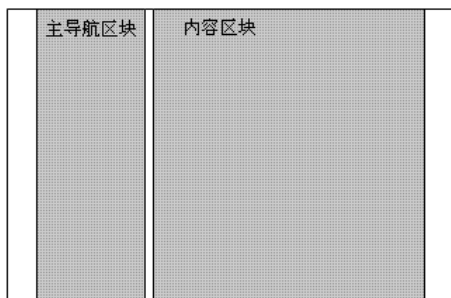


图 8.14 两列浮动设计



8.8.3 设置三列浮动的布局

要设置三列浮动的布局，只需要在上例的基础上，在右部内容区的盒子区块中再嵌套两个新的div盒子区块即可。一个盒子区块用于主要内容，设置向左浮动；另一个盒子区块用于次要内容，设置向右浮动。代码如下：

```
1 <html>
2   <head>
3     <title>设置三列浮动</title>
4     <style>
5       body{ margin:0; padding:0; text-align:center; }
6       #container { width:966px; margin:0 auto; text-align:left; }
7       #left_main { float:left; width:256px; height:400px;border:1px solid; }
8       #right_content { float:right; width:700px;}
9       #left_box {
10         float:left; /* 设置左部主要内容区块的css布局样式 */
11         width:400px; /* 设置该区块框向左浮动，脱离文档流 */
12         height:400px; /* 设定该区块框的宽度为400像素 */
13         border:1px solid; /* 设定该区块框的高度为400像素，临时设置 */
14       }
15       #right_box {
16         float:right; /* 设定该区块框的边框为1px的直线边框 */
17         width:290px; /* 设置右部次要内容区块的css布局样式 */
18         height:400px; /* 设置该区块框向右浮动，脱离文档流 */
19         border:1px solid; /* 设定该区块框的宽度为290像素 */
20       }
21     </style>
22   </head>
23   <body>
24     <div id="container">
25       <div id="left_main">
26         主导航区块
27       </div>
28       <div id="right_content">
29         <div id="left_box">
30           主要内容区块
31         </div>
32         <div id="right_box">
33           次要内容区块
34         </div>
35       </div>
36     </div>
37   </body>
38 </html>
```

与前面的实例相似，只要设置需要的宽度，然后将主要内容区块框向左浮动，次要内容区块框向右浮动即可，如图 8.15 所示。

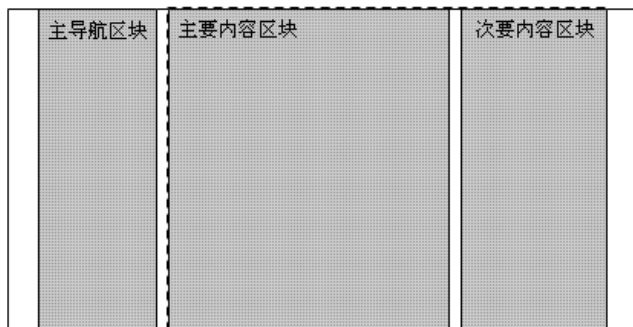


图 8.15 三列浮动设计

8.8.4 设置多列浮动的布局

可以通过借鉴前面介绍的三列浮动布局方式，以在大区块框中再包含小区块框的这种层嵌套的方式实现多列浮动布局。也可以通过设置所有区块框向左浮动，然后在每个区块框之间建立一个垂直隔离带的方式实现多列浮动布局。在下面的示例中，创建一个水平导航条，将所有的菜单项都向左浮动排列成一行，并在菜单项之间设置宽度为 1 像素的隔离带。代码如下：

```

1 <html>
2   <head>
3     <title>设置多列浮动--水平导航菜单</title>
4     <style>
5       body{ margin: 0; padding: 0; text-align: center; }
6       #menu {
7         width: 800px;           /* 声明 id 选择器，用于设置菜单的样式 */
8         margin: 0 auto;        /* 菜单区块的宽度设置为 800px */
9         text-align: left;       /* 菜单区块设置为水平居中 */
10        background: #ccc;       /* 将文本设置回原来的居左 */
11      }
12      #menu ul {
13        float: left;            /* 为菜单条设置一个灰色背景 */
14        margin: 0px;            /* 为了兼容性将列表中原有样式全部清除 */
15        padding: 0px;           /* 设置向左浮动，目的是脱离文档流 */
16        list-style: none;        /* 设置列表外边距为 0 */
17      }
18      #menu ul li {
19        width: 99px;            /* 设置列表内边距为 0 */
20        display: block;         /* 消除列表原有类型 */
21        line-height: 30px;      /* 设置每个菜单项列表的样式 */
22        text-align: center;      /* 设置都向左浮动 */
23      }
24      #menu .menudiv {
25        float: left;            /* 每个菜单项宽度为 99px */
26        width: 1px;             /* 改变为块标记的区块 */
27        height: 20px;           /* 设置行高为 30px，目的是垂直居中 */
28        background: #888;       /* 设置文本水平居中 */
29        margin-top: 5px;
30      }

```



```
31     }
32     </style>
33 </head>
34 <body>
35     <div id="menu">
36         <ul>
37             <li><a href="#">菜单（一）</a></li> <!-- 菜单项的一个链接 -->
38             <li class="menudiv"> </li> <!-- 使用1px的隔离带，分隔两个菜单项 -->
39             <li><a href="#">菜单（二）</a></li>
40             <li class="menudiv"> </li>
41             <li><a href="#">菜单（三）</a></li>
42             <li class="menudiv"> </li>
43             <li><a href="#">菜单（四）</a></li>
44             <li class="menudiv"> </li>
45             <li><a href="#">菜单（五）</a></li>
46             <li class="menudiv"> </li>
47             <li><a href="#">菜单（六）</a></li>
48             <li class="menudiv"> </li>
49             <li><a href="#">菜单（七）</a></li>
50             <li class="menudiv"> </li>
51             <li><a href="#">菜单（八）</a></li>
52         </ul>
53     </div>
54 </body>
55 </html>
```

运行后的结果如图 8.16 所示。

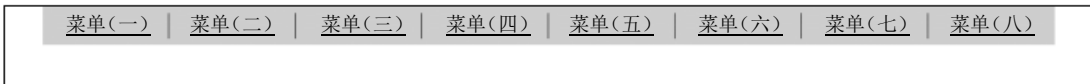


图 8.16 设置多列浮动的菜单实例布局

8.9 DIV+CSS网站首页布局示例

首页的设计直接影响网站的整体形象，虽然没有一个统一的规范，但最好将其设计为大众化的，只要信息内容能够合理地编排即可，使用户可以方便地找到需要的信息。另外，首页的高度最好不要超过 3 个屏幕，页面中使用的颜色最好也不要超过 3 种。通常一个网站首页包含页眉、Logo、banner、主导航菜单、主内容栏目和次内容栏目区块、友情链接和页脚等区块框。本例将实现的效果如图 8.17 所示。

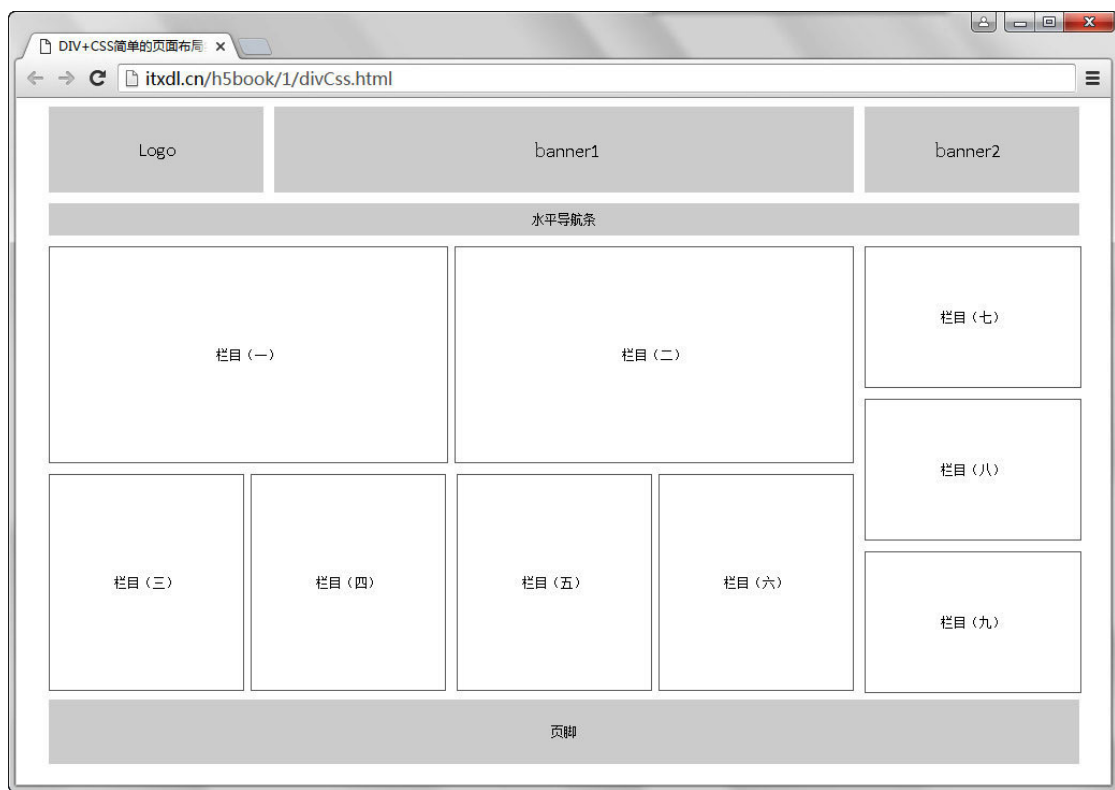


图 8.17 DIV+CSS 布局示例演示效果

8.9.1 HTML文件的设计

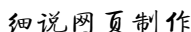
使用 CSS 布局的好处之一就是它能够控制页面布局,而不需要使用过多的 HTML 标签,只需要使用一些<div>、、、之类的标签即可。这样不仅使网站源代码更加简洁,而且能把网页中的主要内容放在前面,使网站的内容完全展现在搜索引擎面前,便于搜索引擎抓取关键内容。本例设计完成一个简单的页面布局,创建一个名为 divCss.html 的网页文件,所需要的 HTML 代码如下:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>DIV+CSS简单的页面布局实例</title>
6     <!-- 链接外部CSS文件 -->
7     <link rel="stylesheet" type="text/css" href="layout.css">
8 </head>
9 <body>
10     <div id="container">
11         <div id="header">
12             <div id="logo" class="bgcolor">LOGO</div>

```

<!-- 最外层 -->
 <!-- 声明页头盒子 -->
 <!-- 声明Logo盒子 -->



8.9.2 CSS文件的设计

```
1 body{ /* 为网页主体内容标记设置样式 */
2     margin: 0; /* 设定网页外部边距值为0, 消除body标记默认值 */
3     padding: 0; /* 设定网页内部边距值为0 */
4     text-align: center; /* 设定网页的内容居中, 在IE中区块框也会居中显示 */
```

```

5     font: 12px Arial, 宋体;           /* 设置网页文字大小12像素Arial或宋体字 */
6 }
7 .border {                           /* 为需要的区块框定义一类的边线样式 */
8     border: 1px solid #888;         /* 四周边线样式为1像素宽度灰色的直线 */
9 }
10 .bgcolor {                          /* 为需要的区块框定义一类的背景颜色 */
11     background: #DDD;              /* 定义使用该类的区块框背景颜色为淡灰色 */
12 }
13 #container {                       /* 为布局的最外层容器区块框使用id选择器设置样式 */
14     width: 960px;                  /* 设置容器的宽度为960像素 */
15     margin: 0 auto;               /* 设置容器外部边距都为0, 使用auto在Firefox中居中 */
16 }
17 #header {                          /* 为页面中ID为header的页眉区块框容器定义样式 */
18     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
19     width: 100%;                  /* 设置该区块框的宽度和上一层容器区块的宽度相同 */
20 }
21 #logo {                            /* 为页面中ID为logo的Logo区块框容器定义样式 */
22     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
23     width: 200px;                 /* 设置该区块框的宽度为200像素 */
24     height: 80px;                 /* 设置该区块框的高度为80像素 */
25 }
26 #banner {                          /* 为页面中ID为banner的Banner区块框容器定义样式 */
27     float: right;                 /* 设置该区块向右漂浮使其脱离页面的文档流 */
28     width: 750px;                 /* 设置该区块框的宽度为750像素 */
29 }
30 #banner #left {                    /* 为页面中ID为banner的内部ID为left的容器定义组合样式 */
31     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
32     width: 540px;                 /* 设置该区块框的宽度为540像素 */
33     height: 80px;                 /* 设置该区块框的高度为80像素 */
34 }
35 .nav {                             /* 设置一个空白条添加到列表区块的下方 */
36     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
37     height: 10px;                 /* 设置空白条的高度为10像素 */
38     width: 100%;                 /* 设置空白条的宽度填满整个容器 */
39     overflow: hidden;             /* 在IE中最小高度为18像素, 将超出部分隐藏 */
40     clear: both;                  /* 清除该区块框两边的浮动区块 */
41 }
42 #banner #right {                   /* 为页面中ID为banner的内部ID为right的容器定义组合样式 */
43     float: right;                 /* 设置该区块向右漂浮使其脱离页面的文档流 */
44     width: 200px;                 /* 设置该区块框的宽度为200像素 */
45     height: 80px;                 /* 设置该区块框的高度为80像素 */
46 }
47 #menu {                            /* 为页面中ID为menu的菜单区块框容器定义样式 */
48     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
49     width: 100%;                  /* 设置该区块框的宽度和上一层容器区块的宽度相同 */
50     height: 30px;                 /* 设置该区块框的高度为30像素 */
51 }
52 #sidebar {                         /* 为页面中ID为sidebar的右部边条区块框容器定义样式 */
53     float: right;                 /* 设置该区块向右漂浮使其脱离页面的文档流 */
54     width: 200px;                 /* 设置该区块框的宽度为200像素 */
55     height: 410px;                /* 设置该区块框的高度为410像素 */
56 }
57 #sidebar .bar {                    /* 为页面中ID为sidebar的内部类为bar的容器定义组合样式 */
58     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
59     width: 100%;                  /* 设置该区块框的宽度和上一层容器区块的宽度相同 */
60     height: 130px;                /* 设置该区块框的高度为130像素 */
61 }
62 #content {                         /* 为页面中ID为content的主内容区块框容器定义样式 */
63     float: left;                  /* 设置该区块向左漂浮使其脱离页面的文档流 */
64     width: 750px;                 /* 设置该区块框的宽度为750像素 */

```



```
65 }
66 #content .left_box {           /* 为页面中ID为content的内部类为left_box的容器定义组合样式 */
67     float:left;                 /* 设置该区块向左漂浮使其脱离页面的文档流 */
68     width:370px;                /* 设置该区块框的宽度为370像素 */
69     height:200px;              /* 设置该区块框的高度为200像素 */
70 }
71 #content .right_box {          /* 为页面ID为content的内部类为right_box的容器定义组合样式 */
72     float:right;               /* 设置该区块向右漂浮使其脱离页面的文档流 */
73     width:370px;               /* 设置该区块框的宽度为370像素 */
74     height:200px;              /* 设置该区块框的高度为200像素 */
75 }
76 #content .left {               /* 为页面中ID为content的内部类为left的容器定义组合样式 */
77     float:left;                /* 设置该区块向左漂浮使其脱离页面的文档流 */
78     height:200px;              /* 设置该区块框的高度为200像素 */
79     width:180px;               /* 设置该区块框的宽度为180像素 */
80 }
81 #content .right {              /* 为页面中ID为content的内部类为right的容器定义组合样式 */
82     float:right;               /* 设置该区块向右漂浮使其脱离页面的文档流 */
83     height:200px;              /* 设置该区块框的高度为200像素 */
84     width:180px;               /* 设置该区块框的宽度为180像素 */
85 }
86 #footer {                      /* 为页面中ID为footer的页脚区块框容器定义样式 */
87     float:left;                /* 设置该区块向左漂浮使其脱离页面的文档流 */
88     width:100%;                /* 设置该区块框的宽度和上一层容器区块的宽度相同 */
89     height:60px;               /* 设置该区块框的高度为60像素 */
90 }
```

本章小结

DIV+CSS 布局页面的优势：表现和内容相分离、代码简洁、提高页面浏览速度、易于维护和改版、提高搜索引擎对网页的索引效率。每个 HTML 元素都可以看作一个区块，类似于装了东西的盒子，称为盒子模型。一个盒子是装内容的区块，如果多个盒子组合在一起使用，再通过定位和浮动进行设置，就可以对整个页面进行布局。标准 W3C 盒子模型的范围包括 margin、border、padding、内容区，并且内容区部分不包含其他部分。DIV+CSS 进行页面布局时需要处理的兼容性问题有很多，通常在进行页面布局调试时将其划分为 IE 和非 IE 两类。本章的学习建议：要多写、多练、多布局一些有个性的页面；把书中的每个例子输入到计算机中进行实践操作；对自己布局的页面不断提出更高的要求。

本章习题

1. 关于基本 CSS 代码书写规范，不正确的是（ ）。
- A. 尽量不缩写
- B. 全部小写，且每一项 CSS 定义写成一行

C. ID 必须是唯一的, 且用在结构的定义中

D. CSS 可以尽量使用 expression

2. 用来标示有序列表的标签是 ()。

A.

B.

C.

D. <dl>

3. 下列哪句语句是 input 类型中用于添加可单击按钮的? ()

A. <input type="submit">

B. <input type="image">

C. <input type="button">

D. <input type="reset">

4. 下列哪个样式定义后, 内联 (非块状) 元素可以定义宽度和高度? ()

A. display:inline

B. display:none

C. display:block

D. display:inherit

5. 下述有关 CSS 属性 position 的属性值的描述, 说法错误的是 ()。

A. static: 没有定位, 元素出现在正常的流中

B. fixed: 生成绝对定位的元素, 相对于父元素进行定位

C. relative: 生成相对定位的元素, 相对于元素本身正常位置进行定位

D. absolute: 生成绝对定位的元素, 相对于 static 定位以外的第一个祖先元素进行定位

6. 下列选项中, 应用了行内样式的是 ()。

A. <p class="style">

B. <p style="color:red">

C. <p id="content">

D. <p class="style1 style2">

7. 关于下列代码片段, 分析不正确的是 ()。

```
<style type="text/css">
```

```
a {
```

```
display:inline;
```

```
display:block;
```

```
width:100px;
```

```
height:30px;
```

```
border:1px solid red;
```

```
}
```

```
</style>
```

A. 以上代码用于修改超链接标签的默认样式

B. 超链接将以块状方式显示

C. 超链接的宽度为 100px, 高度为 30px

D. 超链接在同一行显示

8. 下列关于标签<div>与的说法, 正确的是 ()。

A. 标签只能用于文本内容

B. <div>标签显示时将独占一行

C. 标签在浏览器中显示时将占满一行

D. 多个<div>标签元素将在同一行显示



9. 在制作系统后台页面时，通常会优先考虑的布局是（ ）。
- A. DIV 布局 B. 框架布局 C. 表格布局 D. 文字布局
10. 采用 DIV+CSS 设计页面布局的优势不包括（ ）。
- A. 减少页面冗余代码 B. 更容易修改和维护
- C. 容易被搜索引擎收录 D. 可以轻松控制页面布局
11. 简述题：简单介绍一下 CSS 的盒子模型。



本章习题及其答案



本章资源包



本章扩展知识

第9章

响应式布局



不管用户使用何种终端访问网站，都能够自动识别适应终端设备的分辨率及宽度，从而使网站可以在众多设备中无缝浏览（在CSS2.1中定义了各种媒体类型，包括显示器、便携设备、电视机等）。随着智能手机和平板电脑的逐渐普及，普通的网站对于这些持有移动设备的用户来说，访问无疑是困难的，他们必须在设备上放大和缩小整个网页，以便能够使用合适的字体大小进行浏览，稍不注意就可能会点错进入其他区域。这种状况在响应式网页设计中会有所改善。响应式布局是指一个网站能够兼容多个终端，而不是为每个终端做一个特定的版本。这个概念是为解决移动互联网浏览而诞生的，响应式布局可以为不同终端的用户提供更加舒适的界面和更好的用户体验。而且随着大屏幕移动设备的普及，可以说响应式布局是大势所趋，现在越来越多的网站开始采用响应式的布局方案。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

9.1

响应式布局的优缺点

使用响应式布局的优点比较明显，即面对不同分辨率的设备灵活性强，以及能够快捷解决多设备显示适应问题。也就是说，在非响应式Web设计中，多设备中访问视觉不统一，非最佳视觉，而在响应式设计中能达到多终端视觉和操作体验风格统一，并且可以做到兼容



当前和未来设备。另外，响应式 Web 设计中的大部分技术都可以用在 Web APP 开发中，这也是现在比较流行的开发模式。使用响应式布局还可以节约开发成本，维护起来也轻松很多。

当然，响应式布局也存在缺点。比如，兼容各种设备工作量大，效率低下；代码累赘，会出现隐藏无用的元素，加载时间加长（相比手机定制网站流量稍大，但相对于加载一个完整的 PC 端网站显然要小得多）；受多方面因素影响而达不到最佳效果；在一定程度上改变了网站原有的布局结构，会出现用户混淆的情况。所以，是否在产品中使用响应式布局方案可以折中考虑。

9.2 如何设计响应式布局

一个普通的自适应显示的三栏网页，当用不同的终端来查看这个页面时，会根据几种终端来显示不同的样式。在电脑上是三列，在 iPad 上也应该是三列，在大屏手机上是一行，在屏幕小于 320px 的手机上只显示主要内容，隐藏了次要元素。

我们知道，在不同的设备中，浏览器的窗口尺寸可能是不同的。如果只针对某种窗口尺寸来制作网页，在其他设备中呈现该网页时就会产生很多问题；如果针对不同的窗口尺寸制作不同的网页，则要制作的网页就会太多。为了解决这个问题，在 CSS3 中加入了 Media Queries 模块（媒体查询），它是制作响应式布局的一个利器，使用这个工具我们可以非常方便、快捷地设计出各种丰富的、实用性强的界面。在 Media Queries 模块中允许添加媒体查询表达式，用以指定媒体类型，然后根据媒体类型选择应该使用的样式。换句话说，允许在不改变内容的情况下，在样式中选择一种页面的布局以精确地适应不同的设备，从而改善用户体验。网页制作者只需要针对不同的浏览器窗口尺寸来编写不同的样式，然后让浏览器根据不同的窗口尺寸来选择使用不同的样式即可。

到目前为止，Media Queries 模块得到了 Firefox、Safari、Chrome 及 Opera 浏览器的支持。移动终端上一般都是对 CSS3 支持比较好的高级浏览器，不需要考虑响应式布局的媒体查询兼容问题。IE 8 及以下版本浏览器不支持媒体查询，虽然也有很多种解决方案，几乎都是通过下载 JavaScript 插件来实现，但没有必要去解决这个问题，因为小屏幕显示都是在移动端使用，IE 低版本只要按正常 PC 端页面显示页面布局即可。

在开发中只要拖动浏览器就可以触发判断条件，测试的时候不需要去找一堆手机，只要把自己的浏览器窗口缩小到一定尺寸就可以了。另外，还有一个不错的在线 Web 工具——Responsivator（开源的，直接从 <https://github.com/johnpolacek/Responsivator> 下载），里面提供了很多不同尺寸的屏幕的展示效果，只需要提供一个 URL，就可以看到网站在不同屏幕中的显示效果。

9.3 响应式布局实例

在学习 Media Queries 模块前,先通过一个响应式布局实例来了解一下响应式布局和 Media Queries 模块的简单应用。在本例中,使用 HTML5 的结构元素定义了 5 个盒子。当浏览器窗口尺寸不同时,页面会根据当前窗口的大小选择使用不同的样式。当窗口宽度在 1024px 以上时,页头和页脚分别在页面的最上方和最下方整行显示,中间主体分为三列显示;当窗口宽度在 640px 以上、1024px 以下时,中间的第三列隐藏;而当窗口宽度在 640px 以下时,5 个区块从上到下排列显示。代码如下:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8" />
5   <title>响应式布局 (Media Queries模块) </title>
6   <style>
7     body,h2{ margin: 0; padding: 0; color: white; }
8     #main, header, aside, footer{ background: #333; margin: 5px 0; }
9     /*全局的样式,不指定屏幕区域使用全局样式,指定的屏幕可以继承并修改这一部分*/
10    h2{ text-align: center; font-size: 3em; }
11    /* 整体容器 宽度960px 居中 */
12    #container{ width: 960px; margin: 0 auto; }
13    /* 页头 宽度100% 行高100px */
14    header{ float: left; width: 100%; line-height: 100px; }
15    /* 左边条 宽度200px 行高400px */
16    #left{ float: left; width: 200px; line-height: 400px; }
17    /* 主体区 宽度540px 行高400px */
18    #main{ float: left; width: 540px; line-height: 400px; margin-left: 10px; }
19    /* 右边条 宽度200px 行高400px */
20    #right{ float: right; width: 200px; line-height: 400px; }
21    /* 页脚 宽度100% 行高80px */
22    footer{ float: left; width: 100%; line-height: 80px; }
23
24    /* 屏幕分辨率在1024px以上使用的样式,按电脑屏幕显示模块 */
25    @media screen and (max-width: 1024px){
26      h2{ text-align: center; font-size: 3.5rem; color: yellow; }
27      #container{ width: 960px; margin: 0 auto; }
28      header{ float: left; width: 100%; }
29      #left{ float: left; width: 200px; }
30      #main{ float: left; margin-left: 10px; width: 540px; }
31      #right{ width: 200px; float: right; }
32      footer{ float: left; width: 100%; }
33    }
34
35    /* 屏幕分辨率在640px以上、1024px以下使用的样式,按平板显示模块 */
36    @media screen and (min-width: 640px) and (max-width: 1024px){
37      h2{ text-align: center; font-size: 2.5rem; color: #f0f; }
38      #container{ width: 600px; margin: 0 auto; }
39      #left{ float: left; width: 160px; }
40      #main{ float: left; margin-left: 10px; width: 430px; }
41      /* 次要区块可以不在屏幕上显示 */

```




```
42     #right{ display: none; }
43 }
44
45 /* 屏幕分辨率在640px以下使用的样式，从上往下排列5行显示 */
46 @media screen and (max-width: 639px){
47     h2{ text-align: center; font-size: 3.5rem; color: #0f0; }
48     #container{ width: 400px; margin: 0 auto; }
49     #left{ float: left; width: 100%; line-height: 100px; }
50     #main{ float: left; margin-left: 0; width: 100%; line-height: 200px;}
51     #right{ width: 100%; float: left; line-height: 100px; }
52 }
53
54 </style>
55 </head>
56 <body>
57     <section id="container">
58         <header>
59             <h2>header</h2>
60         </header>
61         <aside id="left">
62             <h2>left</h2>
63         </aside>
64         <section id="main">
65             <h2>main</h2>
66         </section>
67         <aside id="right">
68             <h2>right</h2>
69         </aside>
70         <footer>
71             <h2>footer</h2>
72         </footer>
73     </section>
74 </body>
75 </html>
```



在本例中，通过不同的屏幕分辨率访问可以获取不同的样式，在各自的样式中重新设置了每个区块的布局，不仅需要改变布局样式，在不同的屏幕分辨率下，字体、图片及背景图片同样需要重新设置样式，以适应当前屏幕下的内容展示。至于要判断多少种分辨率，完全取决于产品的需求。常见的分辨率种类有手机、平板电脑（这些终端存在横屏、竖屏的区别）、桌面显示器，一般大于 960px 的显示器都可以采用默认样式而不必在媒体查询里判断。本例在不同屏幕分辨率下的展示效果如图 9.1～图 9.3 所示。

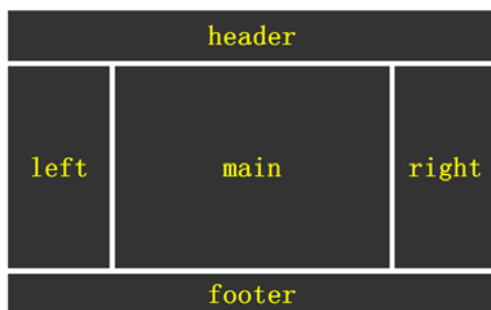


图 9.1 窗口宽度在 1024px 以上时的页面显示

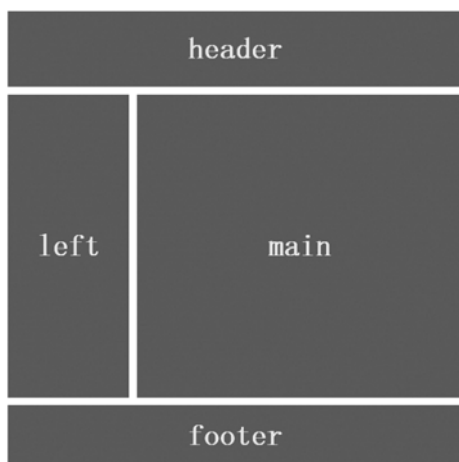


图 9.2 窗口宽度在 640px 以上、1024px 以下时的页面显示



图 9.3 窗口宽度在 640px 以下时的页面显示

在不同窗口大小的子样式区域中，可以继承全局的样式，只要重新设置需要改变的样式即可。另外，为了页面摆放合适，可以将一些次要的区块内容隐藏。如在图 9.2 中，窗口宽度为 640~1024px 时，将右部区块隐藏。

9.4 Media Queries模块的使用方法

在上例中，我们使用 Media Queries 模块来根据 3 种不同尺寸的窗口使用 3 种不同的样式。通过不同的媒体类型和条件定义样式表规则，媒体查询让 CSS 可以更精确地作用于不



同的媒体类型和同一媒体的不同条件。媒体查询的大部分媒体特性都接受 `min` 和 `max` 用于表达“大于或等于”和“小于或等于”。例如，`width` 会有 `min-width` 和 `max-width` 媒体查询可以被用在 CSS 中的 `@media` 和 `@import` 规则上，也可以被用在 HTML 和 XML 中。通过这个标签属性，我们可以很方便地在不同的设备上实现丰富的界面，特别是在移动设备上，媒体查询将会运用得更加广泛。媒体查询能够获取的值如下：

- 设备的宽和高（`device-width`，`device-height`），主要是显示屏幕或触觉设备。
- 渲染窗口的宽和高（`width`，`height`），主要是显示屏幕或触觉设备。
- 获取设备的手持方向，如是横向还是竖向（`orientation(portrait|landscape)`），以及打印机设备等。
- 获取画面比例（`aspect-ratio`），包括点阵打印机等。
- 获取设备比例（`device-aspect-ratio`），包括点阵打印机等。
- 获取对象颜色或颜色列表（`color`，`color-index`），也包括显示屏幕。
- 获取设备的分辨率（`resolution`）。

9.4.1 语法结构及用法

媒体查询有两种使用方式：一种是在 CSS 样式中内嵌“`@media`”，在同一个 CSS 中通过不同窗口编写不同的样式去选择；另一种是使用外部样式表的引用，在 `@import` 和 `link` 中使用“`@media`”，根据不同的窗口大小选择对应的样式文件。这两种方式的使用方法是一样的。Media Queries 模块的使用方法如下：

```
@media 设备类型 only (选取条件) not (选取条件) and (设备特性),设备二 { 样式代码 }
```

在 CSS3 的 Media Queries 模块中支持对外部样式表的引用，使用方法如下：

```
@import url(color.css) screen and (min-width: 1000px); /*使用@import 导入 CSS 文件*/
```

或

```
<link rel="stylesheet" type="text/css" media="only screen and (max-width: 480px),only screen and (max-device-width: 480px)" href="link.css" rel="external nofollow" /> /*使用 link 导入外部 CSS 文件*/
```

简写：

```
<link rel="stylesheet" type="text/css" media="screen and (max-width: 480px)" href="link.css" />
```

在上例中 `only` 可省略，限定于计算机显示器，第一个条件 `max-width` 是指渲染界面的最大宽度，第二个条件 `max-device-width` 是指设备最大宽度。在样式表中内嵌 `@media` 的代码示例如下：

```
@media (min-device-width:1024px) and (max-width:989px), screen and (max-device-width:480px), (max-
```

```
device-width: 480px) and (orientation:landscape), (min-device-width:480px) and (max-device-width:1024px) and (orientation:portrait) { 样式代码 }
```

简写：

```
@media screen and (max-width:640px) { 样式代码 }
```

在上面的示例代码中，设置了计算机显示器分辨率（宽度）大于或等于 1024px（并且最大可见宽度为 989px），屏宽在 480px 及其以下手持设备，屏宽在 480px 及横向（该尺寸平行于地面）放置的手持设备，屏宽大于或等于 480px、小于 1024px 及垂直放置设备的 CSS 样式。从上面的例子中可以看出，字符间以空格相连，选取条件包含在小括号内，样式代码为兼容设置的样式表，包含在花括号里面。only（限定某种设备，可省略）、and（逻辑与）、not（排除某种设备）为逻辑关键字，多种设备用逗号分隔，这一点继承了 CSS 的基本语法特性。

9.4.2 可用的设备类型

在上面的语法中，例如在 CSS 中嵌入“@media”的方式，开头必须书写“@media”，然后指定设备类型（上例使用 screen 代表计算机显示器）。CSS 中定义了 10 种设备类型，可以指定的值与该值所代表的设备类型如表 9.1 所示。

表 9.1 在 Media Queries 模块中可以指定的值与该值所代表的设备类型

可以指定的值	设备类型
All	所有设备
Braille	盲文，即盲人用点字法触觉回馈设备
Embossed	盲文打印机
Handheld	手持的便携设备
Print	文档打印用纸或打印预览视图模式
Projection	各种投影设备
Screen	彩色计算机显示器屏幕
Speech	语言和音频合成器
Tty	固定字母间距的网格的媒体，比如电传打字机和终端
Tv	电视机类型的设备



9.4.3 可用的设备特性参数

通过设备类型可以区分使用的设备，再通过设备特性参数来设置同一设备的不同型号。例如，通过设备类型指定计算机显示器，再通过设备特性参数指定使用多大屏幕的显示器。设备特性的书写方式与样式的书写方式很相似，分为两个部分，由冒号分隔，冒号前书写设备的某种特性，冒号后书写该特性的具体值，如“(min-width:320px)”。CSS 中的设备特性共有 13 种，是一个类似于 CSS 属性的集合。但与 CSS 属性不同的是，大部分设备特性的指定接受 min/max 的前缀用来表示大于等于或小于等于的逻辑，以此避免使用“<”和“>”等字符。对于这 13 种设备特性参数的说明，如表 9.2 所示。

表 9.2 13 种设备特性参数的说明

特 性	可指定值	可用媒体类型	是否接受 min/max 前缀	特性说明
width	带单位的长度值 例如：640px	视觉屏幕/触摸设备	允许	定义输出设备中的页面可见区域宽度（单位一般为 px），即浏览器窗口的宽度
height	带单位的长度值 例如：320px	视觉屏幕/触摸设备	允许	定义输出设备中的页面可见区域高度（单位一般为 px），即浏览器窗口的高度
device-width	带单位的长度值 例如：640px	视觉屏幕/触摸设备	允许	定义输出设备的屏幕可见宽度（单位一般为 px），即设备屏幕分辨率的宽度
device-height	带单位的长度值 例如：320px	视觉屏幕/触摸设备	允许	定义输出设备的屏幕可见高度（单位一般为 px），即设备屏幕分辨率的高度
orientation	只能指定两个值： portrait 或 landscape	位图介质类型	不允许	浏览器窗口的方向是纵向还是横向，当窗口高度值大于等于宽度值时，该特性值为 portrait（横向），否则为 landscape（纵向）
aspect-ratio	比例值 例如：16/9	位图介质类型	允许	定义 width 与 height 的比率，即浏览器的长宽比
device-aspect-ratio	比例值 例如：16/9	位图介质类型	允许	定义 device-width 与 device-height 的比率，即设备屏幕长宽比。如常见的显示器比率：4/3、16/9、16/10
color	整数值	视觉媒体	允许	设备使用多少位的颜色值。如果不是彩色设备，则该值等于 0
color-index	整数值	视觉媒体	允许	色彩表中的色彩数

续表

特 性	可指定值	可用媒体类型	是否接受 min/max 前缀	特性说明
Monochrome	整数值	视觉媒体	允许	定义在一个单色框架缓冲区中每像素包含的单色元件个数。如果不是单色设备，则该值等于 0
Resolution	分辨率值 例如：300dpi	位图介质类型	允许	定义设备的分辨率。例如：96dpi、300dpi、118dpcm
Scan	只能指定两个值： progressive 或 interlace	电视类	不允许	定义电视类设备的扫描方式，progressive 为逐行扫描，interlace 为隔行扫描
Grid	只能指定两个值： 0 或 1	栅格设备	不允许	用来查询输出设备是否使用栅格或基于位图。1 代表是，0 代表否

可以使用 **and** 关键字来指定当某种设备类型的某种特性的值满足某个条件时所使用的样式。例如，以下语句指定了当设备窗口宽度小于 640px 时所使用的样式：

```
@media screen and (max-width: 640px) { 样式代码 }
```

可以使用多条语句来将同一个样式应用于不同的设备类型和设备特性中，指定方式类似如下：

```
@media handheld and (min-width:360px), screen and (max-width: 480px) { 样式代码 }
```

可以在表达式中加上 **not** 或 **only** 关键字。**not** 关键字表示对后面的表达式执行取反操作，书写方法类似如下：

```
/* 样式代码将被用在除便携设备之外的其他设备或非彩色便携设备中 */
@media not handheld and (color) { 样式代码 }
/* 样式代码将被用在非彩色设备中 */
@media all and (not color) { 样式代码 }
```

使用 **only** 关键字的作用是让那些不支持 Media Queries 模块但能够读取 Media Type 的设备的浏览器将表达式的样式隐藏起来。例如：

```
@media only screen and (color) { 样式代码 }
```

上面的语句对于支持 Media Queries 模块的设备来说，将能够正确应用样式，就像 **only** 不存在一样。对于不支持 Media Queries 模块但能够读取 Media Type 的设备（例如 IE 8 只支持 “**@media screen**”）来说，由于先读到的是 **only** 而不是 **screen**，将忽略这个样式。对于不支持 Media Queries 模块的浏览器（例如 IE 8 之前的浏览器）来说，无论是否有 **only**，都将忽略这个样式。



9.5 在移动设备上设置原始大小显示

在 iPhone 系列和 iPod Touch 中使用的是 Safari 浏览器，它支持前面介绍的媒体查询表达式。例如，使用 iPhone 320px×480px 的分辨率去访问我们前面的布局示例，却无法得到我们想看到的结果，并不是从上到下排列显示，而是和计算机显示器访问的布局相同。为什么会这样？因为在 iPhone 中使用的 Safari 浏览器在进行页面显示时是将窗口的宽度作为 980px 进行显示的，因为很多网页是按照 800px 左右的标准制作的，所以 Safari 浏览器默认按照 980px 的宽度进行显示，就可以正常显示绝大多数的网页。所以即使已经写好了页面在小尺寸窗口中运行的样式，iPhone 中的 Safari 浏览器也不会使用这个样式，而是选择窗口宽度为 980px 时所使用的样式。所以，我们需要在移动设备上设置原始大小显示和是否缩放的声明。解决方法是在页面的头部<head></head>之间加上如下所示的语句：

```
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
```

或

```
<meta name="viewport" content="width=600px " />
```

<meta>标签的 viewport 属性是在移动设备上设置原始大小显示和是否缩放的声明。该属性可以使用的参数如下。

- width: viewport 的宽度。
- height: viewport 的高度。
- initial-scale: 初始缩放比例。
- minimum-scale: 允许用户缩放到的最小比例。
- maximum-scale: 允许用户缩放到的最大比例。
- user-scalable: 用户是否可以手动缩放。

如果在页面中已经准备好了在小尺寸窗口中使用的样式，并且有可能在 iPhone 或 iPod Touch 中被打开时，不要忘记加入<meta>标签并在标签中写入指定的窗口宽度。其实还可以通过 viewport 把自己“冒充”成更宽的屏幕。

9.6 响应式网站的内容设计

基于响应式开发网站，除了页面的布局是我们设计的重点，网站中显示的图片 and 文字也是我们不能轻视的内容。

9.6.1 响应式图片显示内容设计

真正具有响应性的 Web 设计是完全调整网站以满足访问者的设备。我们需要在响应式布局的页面上传送最佳的、前后联系的图片尺寸。如果是背景图片，我们可以准备多张不同尺寸的图片，然后在各自的媒体查询样式中使用对应的图片背景，再结合 `min-width`、`min-height`、`max-width`、`max-height` 等样式属性来限制窗口最小或最大宽度和高度，模拟响应式及流式布局，从而保证图片不失真。

如果是在 HTML 中加载图片，使图片适应不同分辨率的屏幕，则可以通过设置图片样式的百分比来实现。但是，它不仅包括缩放，同样重要的是减少传送给观众的图片。比如用户是在移动设备中浏览网站，由于网速慢、流量少等因素，访客可能会放弃访问。为了减少这些元素给网站带来的影响，我们可以用更合适的图片替换访问者使用的设备中的数据。在开发时可以通过一些 JavaScript 插件来解决此类问题。

9.6.2 响应式文字显示内容设计

响应式的设计应该秉承“内容优先，移动优先”的原则。我们知道，网页中的内容主要是由文字、图片等元素组成的，那么文字该如何响应呢？排版是响应式 Web 设计最重要的内容之一，如果想让内容在所有尺寸的屏幕上友好地显示，那么绝对有必要为移动设备优化字体。我们可以使用不用单位的标准来实现这个目标，包括 `pixel`（像素）、`em`、百分比或 `rem`。选择一个正确的标准对设计一个可塑的、响应式的交互界面至关重要。

1. pixel

很早之前，`pixel` 就被用作 Web 设计的单位，原因是它本身固有的精密度和准确度。一旦字体被赋予具体的值，就能在所有设备及浏览器中保持相同的大小。虽然这种方法提供了非常精确的大小控制，但它却与我们需要的弹性和响应式相违背。当给一个“父”元素设置尺寸时，通过继承特性就会把尺寸“传递”给“子”元素（这就是把 CSS 命名为“层叠”样式表的原因）。例如，设置一个固定的像素值给 `body` 的 `font-size`，就会把这个值“传递”给你所设计的所有其他元素。这时候，开发人员如果想单独分配样式，就必须手动设置不同的参数去覆盖已存在的样式。因此，基本编辑及调整将需要小心翼翼地对所有预设样式进行修改。这将引发的不只是不方便的问题，还会在各种浏览器和设备上出现显示不友好的问题，其中很多是由于选择字体大小而造成的。很多读者是在特定字体大小的情况下出现阅读障碍的，而这些都是要被防止出现的，尤其是考虑到我们响应式设计的目标是在所有屏幕尺寸和视窗下很友好地显示。如果你的设计采用前卫的方法，那么精确的像素单位并不算是最好的选择。现在，设备有各种各样的屏幕尺寸，也就是不同像素宽高的屏幕。使用一个特定大小



的字体来适应所有的屏幕，与我们的响应式设计是相违背的。

2. em

em 也是一个表示大小的单位，可以定义 font-size 的值大小。举个例子，如果我们创建一个 div 包含 font-size 值为 16px 的文本，那么 1em 就代表 16px，2em 就等于 32px，以此类推。em 在所有浏览器中是可变化的，不需要为每个元素设置值，因为 CSS 具有继承特性，也就是“层叠”。很明显，使用 em 需要考虑其优缺点。尽管使用 em 使得维护网站的成本降低，但也会阻碍开发人员正在寻求控制、精度和可预测性的字体大小。幸运的是，这个问题很容易解决，可以为大部分内容使用相同的计算单位，还需要添加一些简单的文本元素，如 header 和 footer。

3. 百分比

类似 em 单位，百分比也是可变化的、可被继承的。理论上，二者没有很大的区别，都是计算单位。实际上，需要理解二者的区别是自己计划使用哪种计算单位。你当然不想在指定的元素使用 em，而在其他元素使用百分比。例如，CSS 声明 { font-size:100%; }，这样可以覆盖浏览器默认定义或者其他不想要的百分比声明，在这里，可以使用 em 单位。

4. rem

还有另外一个提供弹性字体大小的单位——rem (root Em)。rem 与 em 很相似，不同的是，rem 只定义“父”元素的尺寸。这个重要的不同点可以解决很多出现内嵌套的元素问题。大多数设计师都知道，绝大多数的设计本质上是包含嵌套的元素。然而，现在需要意识到的是，rem 是一个新单位，因此不是所有的桌面浏览器都能支持、解释或者很好地显示它们，至少目前是这种情况。

前面介绍的 4 种字体的计算单位，究竟哪一种更加适合响应式 Web 设计呢？可以确定的是，em 使得字体更加容易缩放和维护。如果你打算使用百分比来调整字体大小，那么你将可能去改变应用于 body 的百分比，而不用作其他操作。维护是一个大问题，需要在设计中多多考虑。

9.7

响应式网站的设计流程

当客户提出产品功能移动化的需求时，虽然响应式站点并不能算是一种纯粹的移动化解决方案，但是，在某些情况下，这种方式是非常值得考虑的。需要先花些心思将原本的网站打造得更具弹性，使其在各种主流移动设备中都拥有尽量优秀的用户体验。响应式网站设计需要考虑以下流程。

第一步：确定需要兼容的设备类型、屏幕尺寸。

通过用户研究，了解用户使用的设备分布情况，确定需要兼容的设备类型、屏幕尺寸。设备类型包括移动设备（手机、平板）和 PC。对于移动设备，设计和实现的时候要注意增加手势的功能。屏幕尺寸包括各种手机屏幕的尺寸（包括横向和竖向）、各种平板电脑的尺寸（包括横向和竖向）、普通计算机屏幕和宽屏。需要考虑的问题是，某个页面进行响应式设计时其适用的尺寸范围有哪些？例如，某搜索网站的搜索结果页面，跨度可以从手机到宽屏，而该首页由于结构过于复杂，想直接迁移到手机上不太现实，不如直接设计一个手机版的首页。结合用户需求和实现成本，对适用的尺寸进行取舍。比如一些功能操作的页面，用户一般没有在手机端进行操作的需求，所以没有必要进行响应式设计。

第二步：制作线框原型。

针对确定下来的几个尺寸分别制作不同的线框原型，需要考虑清楚在不同尺寸下，页面的布局如何变化，内容尺寸如何缩放，功能、内容的删减，甚至针对特殊的环境作特殊化的设计等。这个过程需要设计师和前端开发人员保持密切的沟通。

第三步：测试线框原型。

将图片导入相应的设备进行一些简单的测试，可帮助我们尽早发现可访问性、可读性等方面存在的问题。

第四步：视觉设计。

注意，移动设备的屏幕像素密度与传统计算机屏幕不一样，在设计的时候需要保证内容文字的可读性、控件可单击区域的面积等。

第五步：前端实现。

与传统的 Web 开发相比，响应式设计的页面由于页面布局、内容尺寸发生了变化，所以最终的产出更有可能与设计稿出入较大，需要前端开发人员和设计师多进行沟通。

本章小结

响应式布局可以为不同终端的用户提供更加舒适的界面和更好的用户体验，而且随着目前大屏幕移动设备的普及，用“大势所趋”来形容也不为过。响应式 Web 设计中的大部分技术都可以用在 Web APP 开发中。响应式布局面对不同分辨率的设备灵活性强，能够快速解决多设备显示适应问题。响应式布局可以节约开发成本，方便维护。响应式布局同样也具有缺点，兼容各种设备工作量大，效率低下；代码累赘，会出现隐藏无用的元素，加载时间加长。通过不同的媒体类型和条件定义样式表规则，媒体查询（Media Queries）让 CSS 可以更精确地作用于不同的媒体类型和同一媒体的不同条件。



本章习题

1. Media Queries 不能够获取哪些值? ()
 - A. 设备的宽和高 (device-width, device-height), 显示屏幕/触觉设备
 - B. 渲染窗口的宽和高 (width, height), 显示屏幕/触觉设备
 - C. 设备的手持方向, 如是横向还是纵向 (orientation (portrait|landscape)), 以及打印机等
 - D. 设备的型号和参数
2. <media>标签的哪个参数可以指定渲染界面最大宽度? ()
 - A. max-width B. max-device-width C. max D. orientation
3. <media>标签的逻辑关键字不包括以下哪个值? ()
 - A. only B. and C. or D. not
4. 下列哪项不是响应式布局的优点? ()
 - A. 面对不同分辨率设备灵活性强
 - B. 能够快速解决多设备显示适应问题
 - C. 兼容各种设备工作量小, 效率高
 - D. 响应式设计中能达到多终端视觉和操作体验风格统一
5. 如果想让内容在所有尺寸的屏幕上友好显示, 那么绝对有必要为移动设备优化字体。我们可以使用不用单位的标准来实现这个目标, 有 4 种可选的字体单位分别是 Pixel、百分比、em 和 rem, 其中 () 是提供弹性字体大小的单位。
 - A. Pixel B. 百分比 C. em D. rem



本章习题及其答案



本章资源包



本章扩展知识

第10章

认识和使用 Sass



Sass 是一门 CSS 的扩展性语言。众所周知，CSS 层叠样式表的编写语法不能被称为一门编程语言，因为 CSS 不包括编程语言的基本特性，如变量、数据类型、流程控制语句等。这样前端开发者在编写 CSS 代码时，只能一行一行地码代码，并且这种 CSS 代码重用性低，基本上是一次性的，同样它的扩展性和维护性都较低。于是，Sass 应运而生。Sass 作为 CSS 扩展语言，能够让前端开发者使用编程的思路去编写 CSS 代码，从而极大地提高了前端代码编写效率、重用性、维护性等。那么本章，笔者就来详细讲解一下 Sass 的语法及其基本使用。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

10.1 初识Sass

在 Sass 出现之前，CSS 基本上是设计师的工具，而不是开发者的工具。在开发者眼中，CSS 是一个很麻烦的东西，它没有变量，也没有条件语句，只是一行行单纯的描述，写起来相当费事。



10.1.1 Sass是什么

打开 Sass (sass-lang.com) 的官网, 我们能够看到对“Sass 是什么”的解答, 如图 10.1 所示。

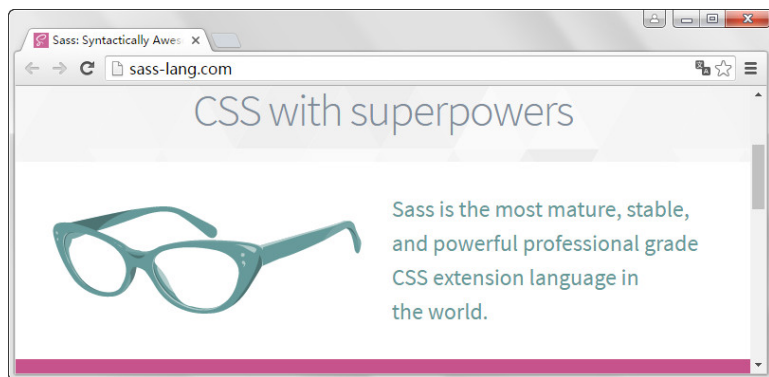


图 10.1 Sass 官网

可以看到, 英文官网上赫然飘出一行大字, 翻译成中文为“具有超级能力的 CSS”; 再看其副标题, 毫不谦逊地告诉所有来访者, “Sass 是世界上最成熟、最稳定、最专业的 CSS 扩展语言”。这种高度的自我评价在业界罕见, 这种高度的自信只有拥有真正实力的应用软件才会有, 不过这些评价对于 Sass 来说还算中肯。为什么呢? 从历史的角度来看, Sass 诞生于 2007 年, 历时 10 年的摸索、打磨、修缮、创新之后, Sass 也逐渐迎合了发展趋势, 降低了学习门槛, 让广大的开发者学习起来简单、易上手, 同时在功能上也表现出其超群、强大的一面, 在我们后面要讲的实例中会让你体会到它的强大。

Sass 从本质上讲, 是一门独立的编程语言, 它具有相关的编程语法, 完全兼容 CSS 语法, 只不过是在 CSS 语法之外添加了一些编程的特性, 如变量、数据类型、函数、混合宏、流程控制语句等。我们使用 Sass 编程来编写的代码, 最终会编译生成 CSS 文件, 因为最终浏览器只认 CSS 文件。

用一句话来概括 Sass 的功能: 我们编写的 Sass 代码在编译之后最终生成的是 CSS 文件。

10.1.2 Sass的作用

在了解完 Sass 之后, 相信大多数人更期望了解它是如何将我们编写 CSS 代码的效率提高的? 是如何让我们的 CSS 代码更具有可维护性的? 是如何让我们编写的 CSS 代码更具有重用性的? 那么笔者就来讲述一个 Sass 编码的实例, 以此来解答这些疑问。

首先我们先拟定一个需求, 让 HTML 中按钮有相同的大小和不同的款式。那么笔者根

据这一需求，使用 Sass 编写出了如下代码：

```

1 $btn-default-color:           #333 !default;
2 $btn-default-bg:              #fff !default;
3 $btn-default-border:          #ccc !default;
4
5 $btn-success-color:           #fff !default;
6 $btn-success-bg:              #ffcccc !default;
7 $btn-success-border:          darken($btn-success-bg, 5%) !default;
8
9 $btn-info-color:               #fff !default;
10 $btn-info-bg:                 #99cccc !default;
11 $btn-info-border:             darken($btn-info-bg, 5%) !default;
12
13 $btn-warning-color:           #fff !default;
14 $btn-warning-bg:              #c96 !default;
15 $btn-warning-border:          darken($btn-warning-bg, 5%) !default;
16
17 $btn-danger-color:            #fff !default;
18 $btn-danger-bg:               #933 !default;
19 $btn-danger-border:           darken($btn-danger-bg, 5%) !default;
20
21 @mixin button-variant($color,$bgcolor,$border){
22     color:$color;
23     background-color:$bgcolor;
24     border:$border;
25 }
26 // 公共样式
27 .btn {
28     display: inline-block;
29     margin-bottom: 0;
30     font-weight: 15px;
31     text-align: center;
32     vertical-align: middle;
33     touch-action: manipulation;
34     cursor: pointer;
35     border: 1px solid transparent;
36     white-space: nowrap;
37     height:20px;
38     width:50px;
39 }
40 // 默认按钮
41 .btn-default {
42     @include button-variant($btn-default-color, $btn-default-bg,
43     $btn-default-border);
44 }
45 // 成功按钮
46 .btn-success {
47     @include button-variant($btn-success-color, $btn-success-bg,
48     $btn-success-border);
49 }
50 // 信息按钮
51 .btn-info {
52     @include button-variant($btn-info-color, $btn-info-bg, $btn-info-border);
53 }

```



```
52 // 警告按钮
53 .btn-warning {
54   @include button-variant($btn-warning-color, $btn-warning-bg,
55     $btn-warning-border);
56 }
57 // 危险按钮
58 .btn-danger {
59   @include button-variant($btn-danger-color, $btn-danger-bg,
60     $btn-danger-border);
61 }
```

首先我们来分析一下上述代码的具体含义。前 20 行，笔者将按钮的背景色、字体颜色、边框线颜色定义成了变量；第 21~25 行，笔者将按钮的主体设置封装成了一个混合宏（与函数概念类似），并接受三个参数，分别为笔者所定义的变量；第 26~39 行，笔者将按钮的基本模型定义了一段 CSS 代码；第 39~59 行，笔者使用类选择器，分别调用了相应混合宏，并传递了三个参数。

上述 Sass 源码编译之后的代码如下：

```
1 .btn {
2   display: inline-block;
3   margin-bottom: 0;
4   font-weight: 15px;
5   text-align: center;
6   vertical-align: middle;
7   touch-action: manipulation;
8   cursor: pointer;
9   border: 1px solid transparent;
10  white-space: nowrap;
11  height: 20px;
12  width: 50px;
13 }
14
15 .btn-default {
16   color: #333;
17   background-color: #fff;
18   border: #ccc;
19 }
20
21 .btn-success {
22   color: #fff;
23   background-color: #ffcccc;
24   border: #ffb3b3;
25 }
26
27 .btn-info {
28   color: #fff;
29   background-color: #99cccc;
30   border: #88c4c4;
31 }
32
33 .btn-warning {
34   color: #fff;
```

```

35 background-color: #c96;
36 border: #c68c53;
37 }
38
39 .btn-danger {
40 color: #fff;
41 background-color: #933;
42 border: #862d2d;
43 }
44
45 /*# sourceMappingURL=demo.css.map */

```

编译后的 CSS 代码与源码进行对比，区别如下。

(1) 变量和混合宏没有在编译后出现。

(2) 混合宏生成对应的代码。

对比之后相信各位读者也有所发现，Sass 的源码从代码上要多于 CSS 代码，这是因为我们考虑了扩展性和维护性。

那么可以想象一下，假如我们需要对颜色进行微调，那么不需要去寻找内部代码，只需要更改头部的变量即可，使之维护性增强；又如，我们需要扩展一个按钮的颜色，那么我们只需要再额外增加三个变量，然后去调用@mixin 混合宏即可；再如，我们在另一个项目中也用到按钮样式，那么我们只需要在相应项目中使用这个 Sass 即可，并且我们可以很方便地对颜色进行微调。

上例展示了 Sass 可维护性、可扩展性强，并且复用性高的特点，也就意味着效率提高了。Sass 的功能不止于此，还有流程控制语句、内置函数、运算符等，这些都能够提高编码效率，这在 10.2 节会做详细介绍。

10.1.3 Sass 的安装

在使用 Sass 之前，我们应该学会如何安装 Sass 的依赖环境，就像 HTML 依赖于浏览器一样。Sass 是使用 Ruby 语言编写的，Sass 的运行依赖于 Ruby 环境，所以在安装 Sass 之前，我们首先要安装 Ruby 环境。接下来，以 Windows 7 的 32 位操作系统环境为例，对 Ruby 环境进行安装，步骤如下。由于 Mac OS X 平台自带 Ruby 环境，因此这里不再赘述。

第一步：到 Ruby 官网下载 Ruby 环境安装包（www.ruby-lang.org/zh_cn），如图 10.2 所示。

需要注意的是，32 位操作系统和 64 位操作系统是有区别的，选择合适的版本进行安装，32 位为 x86，64 位为 x64。

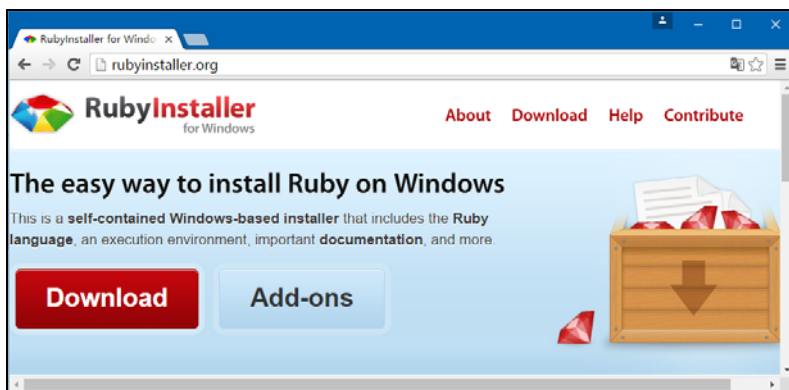


图 10.2 Ruby 下载页面

第二步：安装 Ruby 环境，连续单击“下一步”按钮即可，如图 10.3 所示。

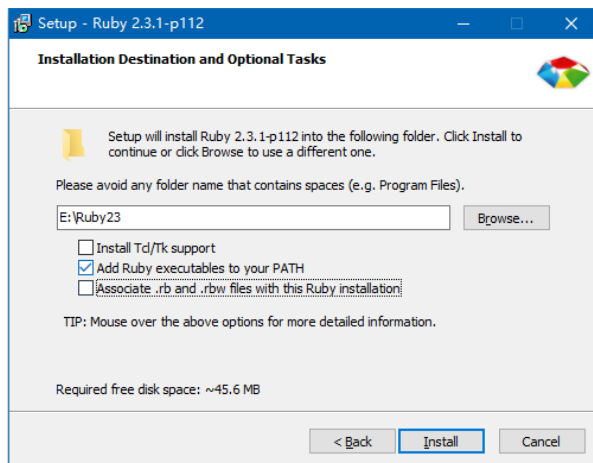


图 10.3 Ruby 安装界面

上步为选择第二项，其含义是自动加入环境变量，这样就能够在 cmd 命令行中使用 Ruby 命令了。

安装完成后，我们使用 Windows+R 快捷键打开 cmd 命令行，输入“ruby -v”查看 Ruby 版本。如果输出了 Ruby 版本，则说明安装成功，效果如图 10.4 所示。

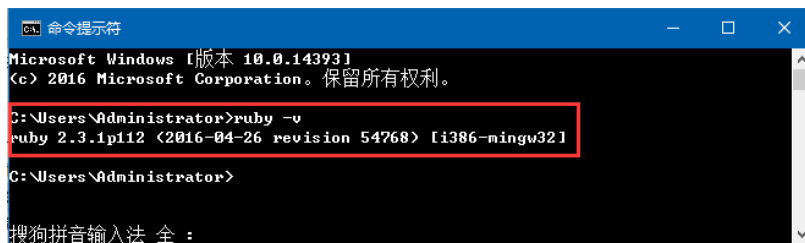


图 10.4 Ruby 安装成功界面

第三步：修改源文件。

Ruby 默认软件包安装工具是 `gem`，与 `composer`（PHP 软件安装工具）、`apt-get`（Ubuntu 软件安装工具）类似，因为 Ruby 是国外开发的，`gem` 的软件源（下载软件的地址）是国外的服务器。受我国的国情限制，一般国外网站不能进行访问，所以我们将源地址修改为中国的镜像地址（镜像与软件源地址包含应用包一致），常用的有腾讯云（<http://gems.ruby-china.org/>）、淘宝镜像（<https://ruby.taobao.org/>）等。这里笔者修改为腾讯云。打开 `cmd` 命令行，输入下面的命令：

```
gem sources --remove https://rubygems.org/
gem sources --add http://gems.ruby-china.org/
gem install sass
```

上述命令的含义是首先删除默认国外的软件源，然后添加腾讯云镜像，最后添加成功后安装 Sass，最终显示如图 10.5 所示。

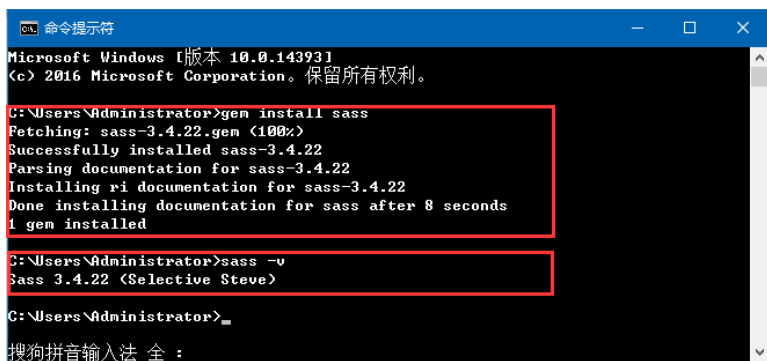


图 10.5 Sass 安装成功界面

到此我们已经成功安装 Sass，接下来就能进行 Sass 的使用和开发了。

10.1.4 Sass的使用和编译

Sass 作为一门具有编程性质的 CSS 扩展语言，具有独立的文件名称，因为其历史原因，它的后缀名可以有两种选择，分别为 `sass` 和 `scss`。其中 `sass` 的编写格式不符合 CSS 编程语法规则，其不能使用大括号和分号，需要严格的缩进来做定界，如下所示：

```
body
  background: #eee
font-size: 12px
p
  background: #0982c1
```

`scss` 完全符合 CSS 编程语法规则，我们可以直接使用 CSS 语法，如下所示：



```
body {  
    background: #eee;  
    font-size: 12px;  
}  
p {  
    background: #0982c1;  
}
```

在前面笔者就反复强调，Sass 只是用来生成 CSS 文件的，那么它是如何生成的呢？是通过 Sass 的编译命令来进行的。最基本的编译命令语法如下：

```
sass 文件名.scss
```

这种编译模式不会生成文件，会在命令行显示编译后的 CSS 代码，如图 10.6 所示。



图 10.6 命令行编译

编译命令包含多种模式，下面统一进行介绍。

1. 生成指定文件名的编译方式

下面这条命令会在当前目录生成一个指定文件名的 CSS 文件名，如下所示：

```
sass 文件名.scss:指定文件名.css
```

效果如图 10.7 所示。

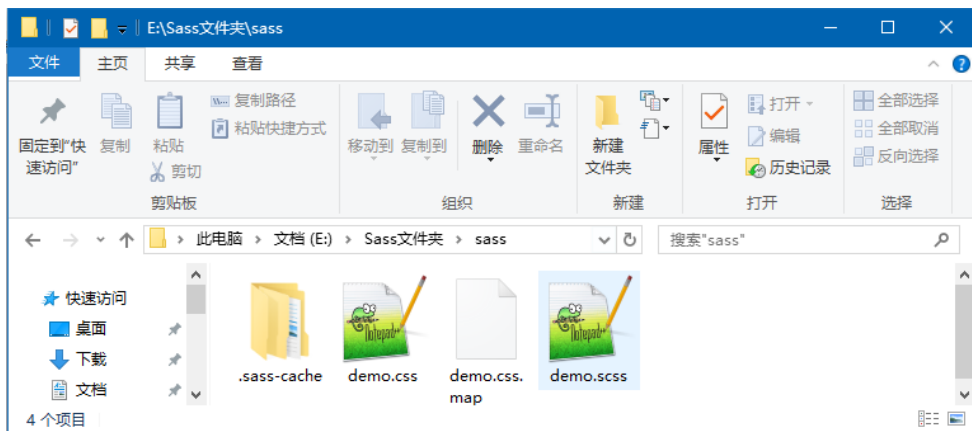


图 10.7 生成指定文件名编译

在图 10.7 中，我们不仅生成了指定的 CSS 文件，而且还生成了两个额外的文件，其中 sass-cache 文件是编译产生的缓存文件，可以删除；demo.css.map 保存的是源文件 demo.scss

和编译后的文件 `demo.css` 对应的关系表，在生产环境中我们只需要使用生成的 CSS 文件即可。

2. 指定文件夹下所有文件编译

你还可以指定编译文件夹，比如创建两个文件夹，分别为 `sass` 和 `css`，那么你只需要执行整个命令就能将指定的文件夹下的文件全部编译到指定的文件夹。命令如下：

```
sass --update sass:css
```

效果如图 10.8 所示。

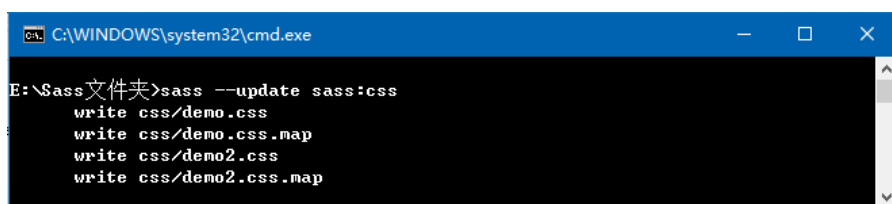


图 10.8 文件夹编译

Sass 编译有 4 种模式，如表 10.1 所示。

表 10.1 编译 Sass 文件的 4 种模式

模 式	效 果
nested	嵌套缩进的 CSS 代码，它是默认值
expanded	没有缩进的、扩展的 CSS 代码
compact	简洁格式的 CSS 代码
compressed	压缩后的 CSS 代码

由此，我们就可以使用编译命令的附加参数 “`--style`” 指定模式来进行编译，如下所示：

```
sass 文件名.scss:文件名.css --style 模式名
```

我们可以来测试一下效果：

```
1 /*这是段落的css样式*/
2 p{
3     border:1px solid red;
4 }
5 /*这是h1标签的css样式*/
6 h1{
7     background-color:green;
8 }
```



由此可以看出，nested 模式仅仅是嵌套了缩进样式，将结尾的花括号提到代码的结尾部分。

```
1 @charset "UTF-8";
2 /*这是段落的css样式*/
3 p {
4     border: 1px solid red; }
5
6 /*这是h1标签的css样式*/
7 h1 {
8     background-color: green; }
9
10 /*# sourceMappingURL=nested.css.map */
```

可以看到，expanded 模式保持原样进行输出。

```
1 @charset "UTF-8";
2 /*这是段落的css样式*/
3 p {
4     border: 1px solid red;
5 }
6
7 /*这是h1标签的css样式*/
8 h1 {
9     background-color: green;
10 }
11
12 /*# sourceMappingURL=expanded.css.map */
```

compact 模式保留了注释，并且将单个选择器的 CSS 样式压缩成了一行。

```
1 @charset "UTF-8";
2 /*这是段落的css样式*/
3 p { border: 1px solid red; }
4
5 /*这是h1标签的css样式*/
6 h1 { background-color: green; }
7
8 /*# sourceMappingURL=compact.css.map */
```

compressed 模式去除了注释，并且将所有代码压缩成了一行。这种模式我们经常用于生产环境，可以有效地避免 CSS 文件过大，导致请求 CSS 资源时间过长的的问题。

```
1 p{border:1px solid red}h1{background-color:green}
2 /*# sourceMappingURL=compressed.css.map */
3
```

这时，如果你感觉还不方便，想达到每修改一次就能够达到即时生成 CSS 样式的效果，那么就需要在编译时加上参数“--watch”，达到监听的效果。命令如下：

```
sass --watch 文件名.scss:文件名.css
```

效果如图 10.9 所示。

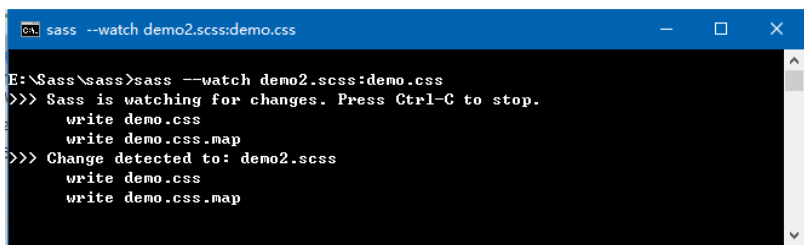


图 10.9 Sass 编译监听

需要注意一点，所有的文件名称都必须是英文，不能是中文。如果目录是中文，则可能会报字符集错误。这时我们对其进行修改，它就会自动监听，然后生成相应的 CSS 文件。

10.2 Sass基本语法与使用实例

本书重点在于 HTML5 与 CSS3 的学习和使用，HTML 与 CSS 不属于编程范畴，更偏向于设计，而 Sass 是编程化的 CSS 的扩展语言，对于没有接触过编程的读者来说可能会有些吃力，这里不对语法做详细介绍，仅作为基本介绍，让读者了解 Sass 的优越性。如果读者对 Sass 感兴趣，可以自行查阅相关资料。

10.2.1 Sass基本语法

在大部分编程语言中，其基本语法都包括这样一些东西，如变量、数据类型、运算符、流程控制语句、函数等，具体这些名词都具有什么功能呢？下面笔者依次进行介绍。

1. 变量

变量，顾名思义，就是可以动态改变的值。可以将它看作一个占位符，开发者可以动态改变这个占位符的值。比如，可以将像素值、颜色值等提炼出来作为一个变量出现。就像下面这样：

```
1 /* CSS格式 */
2 p{
3     background:red;
4     width:100px;
5     height:100px;
6 }
7
8 /* Sass格式 */
9 $background-color:red;
10 $width:100px;
```



```
11 $height:100px;  
12 p{  
13     background:$background-color;  
14     width:$width;  
15     height:$height;  
16 }
```

其实在上例中根本看不出变量的实际效益在哪里，仅仅是解释了其占位符的含义。不过假如网页总体设计时有一个基调，或者说基色，其余网页颜色都由这个基色进行延伸，那么我们只需要定义一个基色，然后对基色进行加深、变浅、反转，就可以得出网页中其他要素的颜色。当我们改变该变量时，网页中所有的颜色都会根据这个基色发生改变。

那么可能有读者问了，如何对基色进行加深、变浅等操作呢？Sass 其实包含大量的内置函数，这些函数可以对颜色执行加深、变浅等操作，那么变量存在的意义就远不止占位符那么简单了，读者还可以对其进行算数运算等操作。

2. 数据类型

什么是数据类型呢？就是在开发中我们能够使用到的值的类型，如数值、字符串、颜色值、布尔值、列表、map 地图类型等。在 CSS 开发中，我们通常使用的就是像素值、颜色值，像素值相当于数值，颜色值也是数值。那么 Sass 的其他数据类型用作什么呢？

布尔值是两个值，即 true 和 false，其含义为真和假，它用于流程控制语句，这里先略过不提，在后面流程控制部分再讲解；列表类型，其实就是一组值，每个值都有一个默认的下标，默认下标自 1 开始依次递增。开发者通过一个方法可以去引用数组中的值，就像下面这样：

```
1 /* 声明一个列表 */  
2 $list-color:red orange yellow green blue black purple;  
3  
4 /* 获取$list-color列表中第一个值 */  
5 nth($list-color, 1);  
6  
7 /* 获取$list-color列表中第七个值 */  
8 nth($list-color, 7);
```

还有一种类型是 map 地图类型，可以说它是升级版的列表，它也保存着一组值，每个值都有对应的一个属性名，开发者可以通过属性名对值进行获取，就像下面这样：

```
1 /* 声明地图数据类型 */  
2 $map-color:(danger:red, warning:yellow, success:green);  
3  
4 /* 获取$map-color指定的值 */  
5 button.danger{  
6     background-color:map-get($map-color, danger);  
7 }  
8  
9 /* 编译后的结果 */  
10 button.danger{  
11     background-color:red;  
12 }
```

3. 运算符

Sass 中的运算符就是算数运算符，在使用算数运算符时，要遵守一个原则，即运算符的前后要留有一个空格，就像下面这样：

```
1 /* 算数运算符前后需要使用空格 */
2 p{
3     height:1px + 2px;
4     width:10px * 10;
5 }
6
7 /* 运算后结果 */
8 p{
9     height:3px;
10    width:100px;
11 }
```

除了算数运算符之外，还有一些其他运算符，包括用于比较的比较运算符，如大于（>）、小于（<）、等于（==）、不等于（!=）、大于等于（>=）、小于等于（<=）等，同时还包括逻辑运算符，如逻辑与（and）、逻辑或（or）、逻辑非（not）。这些运算符常用于流程控制语句之中。

除了以上运算符，还包括字符串连接运算符（+），用来连接两个字符串。在字符串连接时，如果需要进行计算，则需要使用“#{ }”将计算的值括起来，如下所示：

```
1 @charset "UTF-8";
2 $name:"兄弟连";
3 p:before {
4     content: $name + " 已经 #{5 + 5} 周年了!";
5 }
6
7 /* 编译后结果 */
8
9 p:before{
10    content:"兄弟连已经10周年了";
11 }
```

4. 流程控制语句

按正常的代码解析来看，CSS 编码是自上而下依次执行的，而流程控制则是对执行的顺序进行管控，可以控制执行的顺序。Sass 的流程控制语句有 if...else 判断、for 循环、each 循环等，下面笔者依次来讲解一下具体功能。

if...else 语句判断条件是否为真，若条件为真，则执行真区间；若条件为假，则执行假区间。其语法格式如下：

```
@if 条件 {
    真区间;
} @else{
    假区间;
}
```




其中，从`@else`往后均可以省略，那么执行效果就是条件为真，则执行真区间，否则跳过。具体使用请看下例：

```
1 /* 条件控制语句 */
2 @if 1 > 3 {
3     p{
4         border:1px solid red;
5     }
6 } @else {
7     p{
8         border:1px solid blue;
9     }
10 }
11
12 /* 编译后结果 */
13 p{
14     border:1px solid blue;
15 }
```

`@for` 循环也是流程控制语句之一，它可以让一段 CSS 代码重复执行，语法如下：

```
@for 变量 from 开始值 to 结束值 {
    循环体;
}
```

变量相当于一个计数器的占位符，当执行时，变量会被赋值为开始值，然后每循环一次，变量累加 1，直到累加到结束值才会停止。其实“from ... to”可以换成“from ... through”，这两个循环的差别在于，“from ... to”不包含结束值，而“from ... through”则包含结束值。比如“from 1 to 10”会执行 9 次，而“from 1 through 10”则会执行 10 次。举例如下：

```
1 /* 循环语句 */
2 @for $i from 1 through 3 {
3     .li-#{ $i } {
4         width: 2em * $i;
5     }
6 }
7
8 /* 编译后结果 */
9 .li-1 {
10     width: 2em;
11 }
12 .li-2 {
13     width: 4em;
14 }
15 .li-3 {
16     width: 6em;
17 }
```

5. 函数

Sass 包含大量的系统函数，比如将颜色变深的 `darken` 函数、将颜色变浅的 `lighten` 函数、取反色的 `invert` 函数等，这些函数能够帮助我们实现相应的效果，我们只需要使用简单的方法引用即可。举例如下：

```

1 /* 使用系统函数 */
2 $color:red;
3 p{
4     background-color:lighten($color, 20%);
5     border:1px solid invert($color);
6 }
7 p:hover{
8     background-color:darken($color, 20%);
9 }
10
11 /* 编译后结果 */
12 p {
13     background-color: #ff6666;
14     border: 1px solid cyan;
15 }
16 p:hover {
17     background-color: #990000;
18 }

```

当然，除了这些函数，还包括大量的系统函数，这里笔者就不再扩展，读者如果想深入学习，则查阅官方手册即可。

除了自定义函数，你还可以自行创建函数，其效果就类似于系统函数，函数最终会计算出一个数值，用于 Sass 之中。

6. 混合宏

混合宏与函数类似，只不过函数的主要功能是计算，最终返回一个值；而混合宏的主要功能是能够传递参数，并将参数带入混合宏的代码，最终生成的是代码。下面来看一下混合宏的具体语法：

```

@mixin 名称 (参数 1, 参数 2..., 参数 n){
    输出代码;
}

```

那么开发者经常用它来做什么呢？如果你读过 Bootstrap（目前很流行的前端框架）的 Sass 源码，你就知道 Bootstrap 将每个网页中的显式区域都封装成了不同的模块，如按钮模块、表单输入框模块、弹出窗口模块等，那么它到底如何使用呢？举例如下：

```

1 /* 混合宏的声明 */
2 @mixin button($background-color, $color){
3     height:20px;
4     width:100px;
5     color:$color;
6     background-color:$background-color;
7 }
8 /* 混合宏的使用 */
9 .button{
10     @include button(#333, #369);
11 }
12
13 /* 编译后 */

```



```
14 .button {  
15     height: 20px;  
16     width: 100px;  
17     color: #369;  
18     background-color: #333;  
19 }
```

引用方法很简单，使用“@include”调用混合宏，并传递相关参数即可输出相关代码。

10.2.2 Sass使用实例

在了解基本的 Sass 语法之后，笔者给各位读者抛出一个需求，使用 Sass 语法实现七色板，效果如图 10.10 所示。

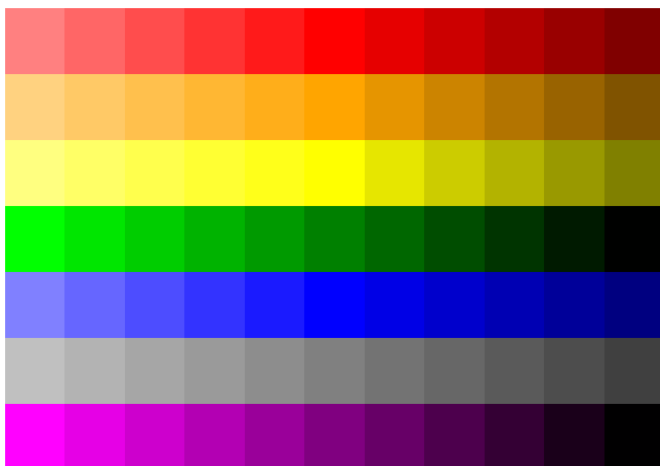


图 10.10 七色板示意图

在初步了解 Sass 之后，大家可能在思路还没有从 CSS 转换过来，那么笔者就带领各位读者一起来分析这个需求的特点。

- 七色板由七种颜色组成，分别为红、橙、黄、绿、蓝、灰、紫。
- 每种颜色都是自左向右逐渐加深。
- 每行格子有 11 个。
- 每行都有一个基色，并且基色的位置位于中间位置，小于基色位置则依次变浅，大于基色位置则依次加深。

既然分析出了特点或需求，那么各位读者就来看看在之前的讲述中是否有哪些基本语法可以为这次需求所用到，笔者将解决方案总结如下。

- 七种颜色可以定义为变量或者列表来解决七色问题。
- 颜色加深函数“darken”和变浅函数“lighten”可以解决颜色问题。

- 流程控制语句中的@for 循环可以解决相同格子与类似行的问题。
- 流程控制语句中的@if 语句可以解决加深和变浅的问题。

既然基本语法能够解决分析出来的每个需求，那么只需要厘清思路就能实现这个七色板。下面笔者就带大家来梳理一下实现的思路。

首先实现前台的 HTML 代码，如下所示：

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>七色板实例</title>
6   <link rel="stylesheet" href="css/color.css">
7 </head>
8 <body>
9   <section class="color-container">
10     <!-- 创建七色板位置图 1 -->
11     <section class="red">
12       <div></div><div></div><div></div><div></div>
13       <!-- 基色位置 -->
14       <div></div>
15       <div></div><div></div><div></div><div></div>
16     </section>
17     <!-- 创建七色板位置图 2 -->
18     <section class="orange">
19       <div></div><div></div><div></div><div></div>
20       <!-- 基色位置 -->
21       <div></div>
22       <div></div><div></div><div></div><div></div>
23     </section>
24     <!-- 创建七色板位置图 3 -->
25     <section class="yellow">
26       <div></div><div></div><div></div><div></div>
27       <!-- 基色位置 -->
28       <div></div>
29       <div></div><div></div><div></div><div></div>
30     </section>
31     <!-- 创建七色板位置图 4 -->
32     <section class="green">
33       <div></div><div></div><div></div><div></div>
34       <!-- 基色位置 -->
35       <div></div>
36       <div></div><div></div><div></div><div></div>
37     </section>
38
39     <!-- 创建七色板位置图 5 -->
40     <section class="blue">
41       <div></div><div></div><div></div><div></div>
42       <!-- 基色位置 -->
43       <div></div>
44       <div></div><div></div><div></div><div></div>
45     </section>
46     <!-- 创建七色板位置图 6 -->

```



```
47 <section class="gray">
48   <div></div><div></div><div></div><div></div><div></div>
49   <!-- 基色位置 -->
50   <div></div>
51   <div></div><div></div><div></div><div></div><div></div>
52 </section>
53 <!-- 创建七色板位置图 7 -->
54 <section class="purple">
55   <div></div><div></div><div></div><div></div><div></div>
56   <!-- 基色位置 -->
57   <div></div>
58   <div></div><div></div><div></div><div></div><div></div>
59 </section>
60 </section>
```

其次，笔者先将七色板分解，仅实现第一行的红色板，如下所示：

```
1 /* 设置基本变量 */
2 $row-width:600px;           // 行宽
3 $row-height:60px;           // 行高
4 $section-count:11;           // 每行格子个数
5
6 /* 设置基本格子属性 */
7 div {
8   width:$row-width / $section-count; // 单个格子宽度
9   height:$row-height;               // 单个格子高度
10  float:left;
11 }
12
13 /* 循环11次，输出十一个格子，并且做相应的加深和变浅判断 */
14 @for $i from 1 through $section-count {
15   .red div:nth-child(#{ $i }) {
16     @if ($i == 6) { // 等于6则输出基色
17       .red div:nth-child(#{ $i }) {
18         background-color: red;
19       }
20     } @else if ($i < 6) { // 小于6则依次变浅
21       .red div:nth-child(#{ $i }) {
22         background-color: lighten(red, (6 - $i) * 5%);
23       }
24     } @else { // 大于6则依次加深
25       .red div:nth-child(#{ $i }) {
26         background-color: darken(red, ($i - 6) * 5%);
27       }
28     }
29   }
30 }
```

可以看到，第一行实现成功。其实第一行实现成功后，仅需要替换对应的选择器和颜色值即可让它实现其余色板。那么，各位读者有没有想到可以传递参数的混合宏呢？

最后，用混合宏封装单行色板，然后使用 for 循环最终实现七色板。最终代码如下：

```

1 @charset "utf-8";           // 设置编码
2 $row-width:600px;           // 行宽
3 $row-height:60px;           // 行高
4 $section-count:11;          // 单行格子个数（必须为奇数）
5 $color:red orange yellow green blue gray purple; // 颜色列表
6 $row-count:length($color);   // 行的个数
7
8 /* 外层容器属性 */
9 .color-container {
10     margin:0 auto;
11     width:$row-width;
12 }
13
14 /* 格子基本属性 */
15 .color-container div{
16     width:$row-width / $section-count;
17     height:$row-height;
18     float:left;
19 }
20
21 /* 七色板单行色板混合宏 */
22 @mixin colors ($selector){
23     @for $i from 1 through $section-count {
24         @if ($i == ($section-count - 1) / 2) {
25             .#{ $selector } div:nth-child(#{ $i }) {
26                 background-color: $selector;
27             }
28         } @else if ($i < ($section-count - 1) / 2) {
29             .#{ $selector } div:nth-child(#{ $i }) {
30                 background-color: lighten($selector, (6 - $i) * 5%);
31             }
32         } @else {
33             .#{ $selector } div:nth-child(#{ $i }) {
34                 background-color: darken($selector, ($i - 6) * 5%);
35             }
36         }
37     }
38 }
39
40 /* 调用混合宏，输出相应CSS代码 */
41 @for $i from 1 through $row-count {
42     @include colors(nth($color, $i));
43 }

```



相信有读者朋友在完成七色板功能之后，还会有这样的疑问：原生的 CSS 就能实现这个功能，为什么还要多学一门语言呢？太麻烦了。

那么笔者抛出几个问题，各位读者请思考一下解决方案。问题如下：

- 如果需要修改格子的尺寸和颜色，是重新做一份，还是修改 Sass 源码？如果修改 Sass 源码，那么该如何修改？
- 如果要将七色板增加成八色板、九色板，那么该如何去扩展？



其实，使用 Sass 编码时正好解决了这些维护、扩展的问题。下面是上述两个问题的回答：

- 如果需要修改尺寸，则仅需要修改行高和行宽即可。格子宽度就是行宽除以单行格子个数。如果修改颜色，则仅需要对颜色列表进行修改即可。
- 如果需要增加或减少色板，则仅需要对颜色列表进行相应的增加或减少，并且与 HTML 代码对应即可。

笔者就不再贴编译后 CSS 代码的图了，它最终使用默认模式生成的代码行数是 250 行左右，而上述的 Sass 代码仅需要 43 行就能完成。

最后，笔者用一句话总结一下 Sass 的优点：使用 Sass 语言编写 CSS 代码，让你的代码具有高质量、高扩展、易维护、高效率的特性。

本章小结

Sass 完全兼容 CSS 语法，并且添加了一些编程的特性，编写的 Sass 代码在编译之后最终生成的是 CSS 文件。Sass 具有维护性高、可扩展性强、复用性高的优点，可以提高我们的编码效率。Sass 是使用 Ruby 语言编写的，Sass 的运行依赖于 Ruby 环境，所以在安装 Sass 之前，我们首先要安装 Ruby 环境。Sass 的后缀名可以有两种选择，分别为 sass 和 scss，其中后缀名为 sass 时需要严格的缩进来做定界。跟大部分编程语言一样，Sass 的基本语法也包括这些基本语法，如变量、数据类型、运算符、流程控制语句、函数等。

本章习题

1. 在 Sass 中编译出来的样式风格有哪些？【多选】（ ）
 - A. 嵌套输出方式 nested
 - B. 展开输出方式 expanded
 - C. 紧凑输出方式 compact
 - D. 压缩输出方式 compressed
2. Sass 的变量不包括以下哪个？（ ）
 - A. 声明变量的符号 “\$”
 - B. 变量名称
 - C. 赋予变量的值
 - D. 变量类型
3. 以下 Sass 代码编译后生成的 CSS 文件为哪项？（ ）

```
a{  
  color: red;  
  &:hover{
```

```
color: green;
```

```
}
```

```
}
```

A. `a{ color: red; } a:hover{ color: green; }`

B. `a{ color: red; } &:{ color: green; }`

C. `a{ color: red; } hover{ color: green; }`

D. `a{ color: red; } a{ color: green; }`

4. SassScript 支持以下哪些数据类型? 【多选】()

A. 数字

B. 文本字符串

C. 颜色

D. 布尔值

5. 所有算数运算都支持颜色值, 并且是分段运算。下面的颜色计算的结果是 ()。

```
p { color: #010203 + #040506; }
```

A. #010203;

B. #040506;

C. #050709;

D. #000000;

6. 简述题: Sass 是什么?



本章习题及其答案



本章资源包



本章扩展知识

第11章

栅格布局



在前面已经介绍了网页中的一种常见布局，即 DIV+CSS。我们通过采用 DIV+CSS 的方式可以实现各种定位和使得网页页面内容与表现相分离。并且通过前面的学习，相信读者也了解了 DIV+CSS 布局页面的优势、如何声明 W3C 的盒子模型、一些和布局有关的 CSS 属性、怎样使用盒子区块的浮动布局和解决了 DIV+CSS 的兼容性问题。

在 DIV+CSS 布局的基础上，本章会为读者介绍另外一种现在流行的布局，即栅格布局。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

11.1 栅格

说到栅格布局，可能有些读者不太清楚。不过说到栅格，脑子里总会有相应的一些画面，如图 11.1 所示。

图 11.1 显示的是以前人们为了保护某部分土地，特意用竹子包围起该块土地。我们把这种编织筑起的竹子围栏就称之为篱笆或栅格。不管之前有没有了解过栅格，相信看了图 11.1，大家会对“栅格”这个词有一个概念，就是用竹、木、铁条等按照一定规律做成的阻拦物。我们可以随意编织不同类型的规律，可以做出菱形、规矩的方形、X 形、圆形等形状。



图 11.1 原始栅格图

不过，在网页中，我们在众多栅格类型规律中，选择其中最经典的一款进行模仿，如图 11.2 所示。



图 11.2 经典款栅格图

之所以称它为栅格中的经典款，是因为它只需要简单的行列垂直水平相交就可以达到既整齐、美观又能起到防御作用的效果。

所以，我们在网页中也可以选择这种通过简单的行列垂直水平相交就可以达到一个页面美观的效果，而且栅格布局还可以很方便地适配 div 盒子所占宽度的比例，我们一般用 `grid` 进行命名。通过栅格布局这种结构，可以方便我们组织内容、易于浏览网站、降低用户端负载。那么现在我们就可以引入两个概念——行（`row`）和列（`column`）。



11.2 盒子模型

在为大家介绍栅格布局原理之前，先来了解一下一般的 HTML 页面布局。在页面布局中依次拥有以下几个板块：页头 header、滚动图片 banner、主体部分 main、页脚 footer。其中页头包含 Logo 及导航栏 nav，主体部分左右排布，分为左侧 sideBar 侧边栏模块和右侧内容 Content 模块，如图 11.3 所示。



图 11.3 页面主要板块

所以，常见的 div 布局代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>盒子模型</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <div class="container"><!-- 整个页面主体 -->
9     <div class="header"><!-- 页面上部页头 -->
10       <div class="nav"><!-- 头部导航栏 -->
11       </div>
12     </div>
13     <div class="banner"></div><!-- 页头下面的一个滚动图片位置 -->
14     <div class="main"><!-- 页面主体 -->
15       <div class="content"><!-- 内容 -->
16       </div>
17       <div class="sideBar"><!-- 侧边栏 -->
18       </div>
19     </div>
20     <div class="footer"></div><!-- 页脚 -->
21   </div>
22 </body>
23 </html>

```

为什么笔者要给大家介绍这个基本的盒子模型呢？是为了让大家了解一个页面中基本的几大板块有哪些或者说基本有哪些元素，大家了解之后自然就会知道该把不同的内容放在各自不同的布局位置上，从而避免犯一些常识性的错误。

11.3 栅格实例

前面介绍了栅格，下面来看一个栅格实例。需求：将某一 div 分为 1:4 的两块，实际效果如图 11.4 所示。



图 11.4 栅格布局



为了这个需求，我们先写一个类名为“content”的父级 div，在父级下有两个子级，这两个子级的类名都不是只有一个，细心的读者会发现有两个。其中一个类名“xdl1”或“xdl2”我们用于控制一些基本属性，如高度、背景色、左浮动；另一个类名“grid1”或“grid2”我们只用于控制模块的宽度。代码如下：

```
1 <html>
2 <head>
3   <title>栅格实战</title>
4   <meta charset="utf-8">
5   <style type="text/css">
6     .content{
7       width:960px;
8       height: 500px;
9       background:rgb(201,226,251);
10      margin: 0 auto;
11    }
12    .xdl1{
13      height: 300px;
14      float: left;
15      background: rgb(112,197,238);
16    }
17    .xdl2{
18      height: 400px;
19      float: left;
20      background: rgb(127,161,226)
21    }
22    .grid1{width:20%;}
23    .grid4{width:80%;}
24  </style>
25 </head>
26 <body>
27   <div class="content">
28     <div class="xdl1 grid1">IT兄弟连</div>
29     <div class="xdl2 grid4">IT兄弟会</div>
30   </div>
31 </body>
32 </html>
```

这种栅格布局可以很方便地适配 div 盒子所占宽度的比例，而且我们一般用 grid 进行命名，如“grid1”和“grid2”。并且，grid 的作用就是在我们完成栅格化后，按比例给 div 块做横向切分，然后设置其宽度。

11.4 Bootstrap框架

之所以在这里突然给大家介绍 Bootstrap 框架，是因为 Bootstrap 内置了一套响应式、移动设备优先的流式栅格系统。该系统不仅可以随着屏幕设备或视口（viewport）尺寸的增加而增加，而且最多会自动分为 12 列。

11.4.1 Bootstrap 现状

Bootstrap，来自 Twitter，是目前很受欢迎的前端框架。Bootstrap 基于 HTML、CSS、JavaScript，简洁灵活，使得 Web 开发更加快捷，是一个 CSS/HTML 框架。Bootstrap 提供了优雅的 HTML 和 CSS 规范，由动态 CSS 语言 Less 写成。Bootstrap 一经推出后颇受欢迎，国内一些移动开发者较为熟悉的框架，如 WeX 5 前端开源框架等，也是基于 Bootstrap 源码进行性能优化而来的。

Bootstrap 中包含了丰富的 Web 组件，根据这些组件，可以快速地搭建一个漂亮、功能完备的网站。其中组件包括：下拉菜单、按钮组、按钮下拉菜单、导航、导航条、路径导航，分页、排版、缩略图、警告对话框、进度条、媒体对象等。

Bootstrap 让前端开发更快速、简单，所有开发者都能快速上手，所有设备都可以适配，所有项目都适用。全球数以百万计的网站都是基于 Bootstrap 构建的，如招聘网、Ghost 中文网、Laravel 中文网等。

11.4.2 栅格系统

Bootstrap 的栅格系统就是通过一系列的行（row）与列（column）的组合创建页面布局，然后把相应的内容放入早已创建好的布局当中。

下面就来简单介绍一下 Bootstrap 栅格系统的工作原理。

栅格系统的实现原理非常简单，仅仅通过定义容器大小，平分为 12 份（也有平分为 24 份或 32 份的，但 12 份最为常见），再调整内外边距，最后结合媒体查询，就制作出了强大的响应式栅格系统。Bootstrap 框架中的栅格系统就是将容器平分为 12 份。

在使用的时候大家可以根据实际情况重新编译 Less（或 Sass）源码（见图 11.5 的第 323 行），以此来修改“12”这个数值（改为 24 或 32，当然也可以分成更多，但不建议这样使用）。

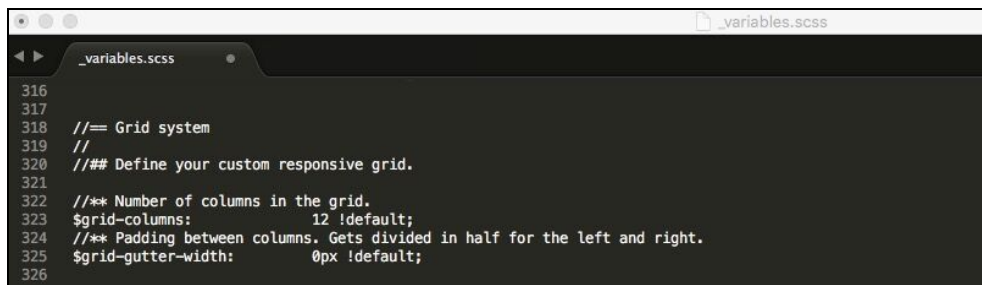


图 11.5 Sass 编译



1. 栅格选项

接下来为读者详细介绍 Bootstrap 的栅格系统。之前介绍了 Bootstrap 的一个特色：支持响应式。所以，Bootstrap 的栅格系统也可以实现在多种不同的移动设备上工作。图 11.6 所展示了 Bootstrap 在不同屏幕设备上工作。

	超小屏幕 手机 (<768px)	小屏幕 平板 (≥768px)	中等屏幕 桌面显示器 (≥992px)	大屏幕 大桌面显示器 (≥1200px)
栅格系统行为	总是水平排列			
.container 最大宽度	None (自动)	750px	970px	1170px
类前缀	.col-xs-	.col-sm-	.col-md-	.col-lg-
列 (column) 数	12			
最大列 (column) 宽	自动	~62px	~81px	~97px
槽 (gutter) 宽	30px (每列左右均有 15px)			
可嵌套	是			
偏移 (Offsets)	是			
列排序	是			

图 11.6 多种屏幕设备

可以看到，在不同的设备上栅格系统行为是不一样的，所以也决定了在不同的设备上样式也会不一样。类名为“container”的容器在设备尺寸 $\geq 1200\text{px}$ 时，会匹配类前缀为“col-lg-”的样式；在 $\geq 992\text{px}$ 同时又 $< 1200\text{px}$ 时，会匹配类前缀为“col-md-”的样式；在 $\geq 768\text{px}$ 同时又 $< 992\text{px}$ 时，匹配类前缀为“col-sm-”的样式；在屏幕尺寸 $< 768\text{px}$ 时，匹配类前缀为“col-xs-”的样式。

下面是 Bootstrap 栅格系统中关键分界点的媒体查询：

```
5 <style type="text/css">
6   /* 超小设备（手机，小于 768px） */
7   /* Bootstrap 中默认情况下没有媒体查询 */
8
9   /* 小型设备（平板电脑，768px 起） */
10  @media (min-width: 768px) { ... }
11
12  /* 中型设备（台式电脑，992px 起） */
13  @media (min-width: 992px) { ... }
14
15  /* 大型设备（大型台式电脑，1200px 起） */
16  @media (min-width: 1200px) { ... }
17
18 </style>
```

2. 基本用法

我们选择栅格系统来布局，其实栅格系统说白了就是行和列的不同组合。在 Bootstrap

框架的栅格系统中提供了 4 种基本的用法，下面以中等屏幕（970px）为例为读者进行介绍。

1) 列组合

列组合就是在 `container` 容器中，首先定义行，然后在每行中再定义不同的列。不同列的类名不一样，决定不同的列所占的父级宽度也不一样。不过，每行的列总和数加起来不能超过 12，如图 11.7 所示。

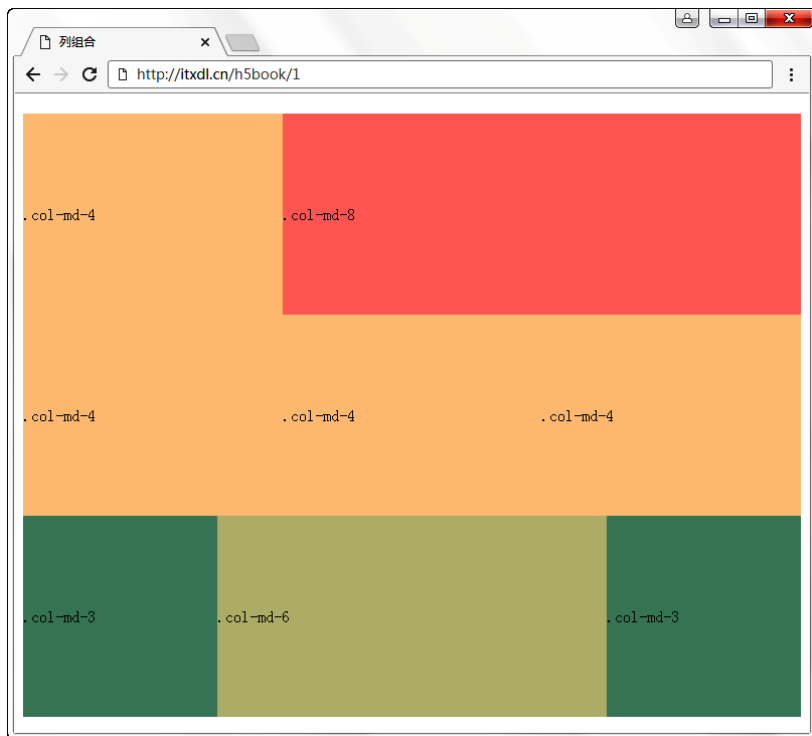


图 11.7 列组合

为了加深读者对行列布局的理解，笔者在这里把行列模块用不同的颜色进行了区分。在第一行中，我们划分了两列，不过两列的布局比例是 4:8；在第二行中，我们划分了三列，每列布局比例是 4:4:4；在第三行中，也划分了三列，每列布局比例是 3:6:3。不知道读者有没有发现，每行列数总和不超过 12。第一行为 $4+8=12$ ，第二行为 $4+4+4=12$ ，第三行为 $3+6+3=12$ 。代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>列组合</title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     .container{position: relative;width:
8       100%;height:300px;background:rgb(247,247,249);}
9     .row{width:100%;background: yellow;margin-top:20px;}
10    .col-md-3,.col-md-4,.col-md-6,.col-md-8{

```




```
10 float: left; height: 200px; line-height: 200px; }
11 .col-md-3{
12     width: 25%;
13     background: rgb(131,175,155);
14 }
15 .col-md-4{
16     width: 33.33333333%;
17     background: rgb(249,205,173);
18 }
19 .col-md-6{
20     width: 50%;
21     background: rgb(200,200,169);
22 }
23 .col-md-8{
24     width: 66.66666667%;
25     background: rgb(252,157,154);
26 }
27 </style>
28 </head>
29 <body>
30 <div class="container">
31     <div class="row">
32         <div class="col-md-4">.col-md-4</div>
33         <div class="col-md-8">.col-md-8</div>
34     </div>
35     <div class="row">
36         <div class="col-md-4">.col-md-4</div>
37         <div class="col-md-4">.col-md-4</div>
38         <div class="col-md-4">.col-md-4</div>
39     </div>
40     <div class="row">
41         <div class="col-md-3">.col-md-3</div>
42         <div class="col-md-6">.col-md-6</div>
43         <div class="col-md-3">.col-md-3</div>
44     </div>
45 </div>
46 </body>
47 </html>
```

给出源代码后，就可以很明显地看出来了。在 container 类的 div 容器里，我们按照行列来划分，又在 row 类的 div 容器里定义不同宽度的列，例如“col-md-4”类的 div 宽度占父级宽度的 33.33333333%，“col-md-6”类的 div 宽度占父级宽度的 50%，而且将所有列皆实现左浮动的布局。

除了上面运用的 3、4、6、8 类，还有其他几种列类名。代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>列组合CSS样式</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7         .container{position: relative; width:
8         100%; height: 300px; background: rgb(247,247,249); }
9         .row{width: 100%; background: yellow; margin-top: 20px; }
10        .col-md-3, .col-md-4, .col-md-6, .col-md-8{
11        float: left; height: 200px; line-height: 200px; }
12        /*确保所有列左浮动*/
13        .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6,
14        .col-md-7, .col-md-8, .col-md-9, .col-md-10, .col-md-11,
15        .col-md-12 {
```

```

15     float: left;
16 }
17 /*定义每个列组合的宽度（使用百分比）*/
18 .col-md-12 {width: 100%;}
19 .col-md-11 {width: 91.66666667%;}
20 .col-md-10 {width: 83.33333333%;}
21 .col-md-9 {width: 75%;}
22 .col-md-8 {width: 66.66666667%;}
23 .col-md-7 {width: 58.33333333%;}
24 .col-md-6 {width: 50%;}
25 .col-md-5 {width: 41.66666667%;}
26 .col-md-4 {width: 33.33333333%;}
27 .col-md-3 {width: 25%;}
28 .col-md-2 {width: 16.66666667%;}
29 .col-md-1 {width: 8.33333333%;}
30 </style>
31 </head>

```

有需求的时候，直接通过类名即可获取相应的 CSS 样式。

2) 列偏移

有时候，在开发过程中，有两个相邻的 div 模块展示内容，写好后给产品经理审核时，产品经理认为这两个相邻的 div 模块靠得太紧了，希望能增大一点宽度。这时候常规思路是使用 margin 或者其他技术手段来实现，但其实我们可以使用 Bootstrap 的列偏移（offset）功能来实现。如何使用？很简单，和上面的列组合一样，只需要在列元素上添加类名“col-md-offset-*”（其中*代表要偏移的列组合数），则具有这个类名的列就会向右偏移。比如，我们在列元素上添加“col-md-offset-2”，表示该列向右移动 2 个列的宽度。图 11.8 所示为案例展示。

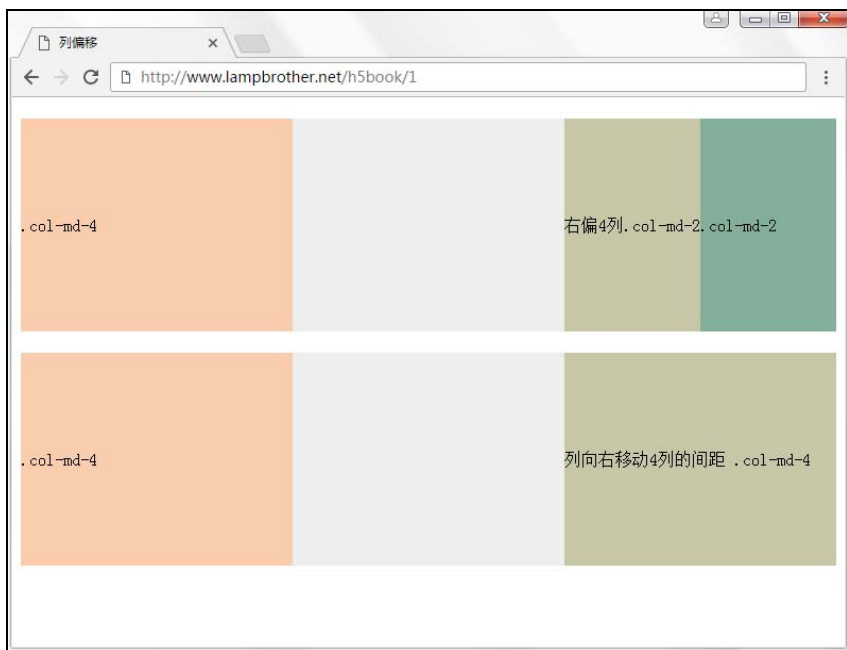


图 11.8 列偏移



图 11.8 是在一个容器（container）里定义两行（row），每行的背景颜色定义为“#eee”，中间显示的空白部分就是行背景色。

在第一行中，实际定义了三列，类名分别是：col-md-4、col-md-2、col-md-2。不过，第二个“col-md-2”的类元素上还添加了一个列类“col-md-offset-4”，使得该 div 模块向右偏移了 4 列的宽度，加上该 div 本身的 2 列宽度，总共占了 6 列宽度。所以，第一行的列总和数是 $4+6+2=12$ ，加起来的列总和数依旧不超过 12。

第二行也是一样，实际上定义了两列，类名分别是：col-md-4、col-md-4。不过，第二个“col-md-4”的类元素上添加了一个类“col-md-offset-4”，使得该 div 模块向右偏移了 4 列的宽度，加上该 div 模块本身的 4 列宽度，所以该 div 模块总共占的宽度是 8 列。因此，第二行的列总和数是 $4+8=12$ ，依旧不超过 12。

源代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>列偏移</title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     .container{position: relative;width: 100%;height:400px;background:#fff;}
8     .row{width:100%;background: #eee;margin-top:20px;height: 200px;}
9     .col-md-2,.col-md-4,.col-md-6,.col-md-8{
10    float: left;height: 200px;line-height:200px;}
11    .col-md-2{
12      width: 16.66666667%;
13      background: rgb(131,175,155);
14    }
15    .col-md-4{
16      width: 33.33333333%;
17      background: rgb(249,205,173);
18    }
19    .col-md-offset-4{
20      margin-left: 33.33333333%;
21      background: rgb(200,200,169);
22    }
23  </style>
24 </head>
25 <body>
26   <div class="container">
27     <div class="row">
28       <div class="col-md-4">.col-md-4</div>
29       <div class="col-md-2 col-md-offset-4">右偏4列.col-md-2</div>
30       <div class="col-md-2">.col-md-2</div>
31     </div>
32     <div class="row">
33       <div class="col-md-4">.col-md-4</div>
34       <div class="col-md-4 col-md-offset-4">列向右移动4列的间距.col-md-4</div>
35     </div>
36   </div>
37 </body>
38 </html>
```

结合前面的分析和源代码，就看得很清晰了。在容器里，想要移动 4 列的距离，而我们总共有 12 列，计算一下这个距离： $4/12=1/3=33.333333\%$ ，所以我们需要用 `margin-left` 去设置移动 33.333333% 的距离；同理可得，想要移动几列，就计算得出占 12 分之几的宽度，再设置相应的 `margin-left` 值。现在，我们可以把其他的偏移类也定义出来：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>列偏移</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7         .container{position: relative;width: 100%;height:400px;background:#fff;}
8         .row{width:100%;background: #eee;margin-top:20px;height: 200px;}
9         /*确保所有列左浮动*/
10        .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6,
11        .col-md-7, .col-md-8, .col-md-9, .col-md-10, .col-md-11,
12        .col-md-12 {
13            float: left;
14        }
15        /*定义每个列组合的宽度（使用百分比）*/
16        .col-md-12 {width: 100%;}
17        .col-md-11 {width: 91.66666667%;}
18        .col-md-10 {width: 83.33333333%;}
19        .col-md-9 {width: 75%;}
20        .col-md-8 {width: 66.66666667%;}
21        .col-md-7 {width: 58.33333333%;}
22        .col-md-6 {width: 50%;}
23        .col-md-5 {width: 41.66666667%;}
24        .col-md-4 {width: 33.33333333%;}
25        .col-md-3 {width: 25%;}
26        .col-md-2 {width: 16.66666667%;}
27        .col-md-1 {width: 8.33333333%;}
28        /*以下设置偏移宽度*/
29        .col-md-offset-12 {margin-left: 100%;}
30        .col-md-offset-11 {margin-left: 91.66666667%;}
31        .col-md-offset-10 {margin-left: 83.33333333%;}
32        .col-md-offset-9 {margin-left: 75%;}
33        .col-md-offset-8 {margin-left: 66.66666667%;}
34        .col-md-offset-7 {margin-left: 58.33333333%;}
35        .col-md-offset-6 {margin-left: 50%;}
36        .col-md-offset-5 {margin-left: 41.66666667%;}
37        .col-md-offset-4 {margin-left: 33.33333333%;}
38        .col-md-offset-3 {margin-left: 25%;}
39        .col-md-offset-2 {margin-left: 16.66666667%;}
40        .col-md-offset-1 {margin-left: 8.33333333%;}
41        .col-md-offset-0 {margin-left: 0;}
42    </style>
43 </head>
44 <body>
45 </body>
46 </html>

```

可以看到，从第 28~41 行设置不同类别的偏移宽度，即从偏移 0 到偏移 100% 的宽度。这样以后大家在使用时就可以直接使用，明确自己想要的移动宽度，然后找到对应的“`col-md-offset-?`”（?代表列数），就可以实现想要设置的移动宽度距离。



读者需要注意一点，使用“col-md-offset-*”对列进行向右偏移时，要保证列与偏移列的总数不超过 12，否则会导致列断行显示。

3) 列排序

列排序，也就是改变列的方向。简单来说就是，通过设置浮动距离来改变左右浮动，在 Bootstrap 的栅格系统中通过添加类名的方式进行功能实现。类名有两个，即“col-md-push-*”和“col-md-pull-*”（其中*代表移动的列组合数）。

下面来看一个简单的例子：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>列排序</title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     .container{position: relative;width:
8     100%;height:300px;background:rgb(247,247,249);}
9     .row{width:100%;background: yellow;margin-top:20px;}
10    .col-md-4,.col-md-8{
11    float: left;height: 200px;line-height:200px;}
12    .col-md-4{
13      width: 33.33333333%;
14      background: rgb(249,205,173);
15    }
16    .col-md-8{
17      width: 66.66666667%;
18      background: rgb(252,157,154);
19    }
20  </style>
21 </head>
22 <body>
23   <div class="container">
24     <div class="row">
25       <div class="col-md-4">.col-md-4</div>
26       <div class="col-md-8">.col-md-8</div>
27     </div>
28   </div>
29 </body>
30 </html>
```

这个例子的演示效果如图 11.9 所示。

可以看到，图 11.9 中左边是类名为“col-md-4”的 div 内容，右边是类名为“col-md-8”的 div 内容。现在如果要互换位置，就需要将左边的“col-md-4”模块向右移动 8 个列的距离，右边的“col-md-8”模块向左移动 4 个列的距离。如何移动呢？在“<div class='col-md-4'>”上添加类名“col-md-push-8”，在“<div class='col-md-8'>”上添加类名“col-md-pull-4”，然后调用样式。

互换位置后的效果如图 11.10 所示。



图 11.9 列排序

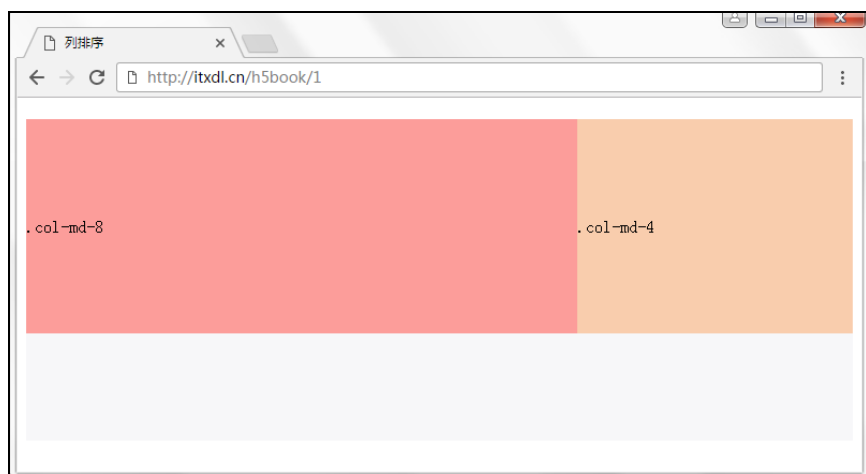


图 11.10 列排序交换

源代码如下：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>列排序-交换</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7         .container{position:relative;width:
8         100%;height:300px;background:rgb(247,247,249);}
9         .row{width:100%;background: yellow;margin-top:20px;}
10        .col-md-4,.col-md-8{
11        float: left;height: 200px;line-height:200px;position: relative;}
12        .col-md-4{
13            width: 33.33333333%;
14            background: rgb(249,205,173);

```



```
14     }
15     .col-md-8{
16         width: 66.66666667%;
17         background: rgb(252,157,154);
18     }
19     .col-md-push-8{
20         left: 66.66666667%;
21     }
22     .col-md-pull-4 {
23         right: 33.33333333%;
24     }
25 </style>
26 </head>
27 <body>
28     <div class="container">
29         <div class="row">
30             <div class="col-md-4 col-md-push-8">.col-md-4</div>
31             <div class="col-md-8 col-md-pull-4">.col-md-8</div>
32         </div>
33     </div>
34 </body>
35 </html>
```

原本在左边的类名为“col-md-4”的模块，添加了类“col-md-push-8”来控制模块的 left 值，想要移动几列，还是计算出占 12 分之几，然后设置 left 值是 12 分之几的距离。在右边的类名为“col-md-8”的模块，在其基础上再添加类“col-md-pull-4”来控制模块的 right 值，依旧是想要移动几列，计算出占 12 分之几，然后设置 right 值是 12 分之几的距离。Bootstrap 就是通过这样添加类的方式设置 left 和 right 来实现定位效果的，实现布局的左右移动。展开来，下面是其余移动距离的类名：

```
1 <!-- 向左 -->
2 .col-md-pull-12 {right: 100%;}
3 .col-md-pull-11 {right: 91.66666667%;}
4 .col-md-pull-10 {right: 83.33333333%;}
5 .col-md-pull-9 {right: 75%;}
6 .col-md-pull-8 {right: 66.66666667%;}
7 .col-md-pull-7 {right: 58.33333333%;}
8 .col-md-pull-6 {right: 50%;}
9 .col-md-pull-5 {right: 41.66666667%;}
10 .col-md-pull-4 {right: 33.33333333%;}
11 .col-md-pull-3 {right: 25%;}
12 .col-md-pull-2 {right: 16.66666667%;}
13 .col-md-pull-1 {right: 8.33333333%;}
14 .col-md-pull-0 {right: 0;}
15
16
17 <!-- 向右 -->
18 .col-md-push-12 {left: 100%;}
19 .col-md-push-11 {left: 91.66666667%;}
20 .col-md-push-10 {left: 83.33333333%;}
21 .col-md-push-9 {left: 75%;}
22 .col-md-push-8 {left: 66.66666667%;}
23 .col-md-push-7 {left: 58.33333333%;}
```

```

24 .col-md-push-6 {left: 50%;}
25 .col-md-push-5 {left: 41.66666667%;}
26 .col-md-push-4 {left: 33.33333333%;}
27 .col-md-push-3 {left: 25%;}
28 .col-md-push-2 {left: 16.66666667%;}
29 .col-md-push-1 {left: 8.33333333%;}
30 .col-md-push-0 {left: 0%;}

```

需要移动几列，相应地选择用“col-md-push-?”或“col-md-pull_?”(?是具体的数字)即可。

4) 列嵌套

前面介绍的布局都是单一地在某一行里定义多个列，其实 Bootstrap 框架的栅格系统还支持列嵌套。我们可以在一个列中添加一个或者多个行(row)容器，然后在这个行容器中插入列，具体插入列的使用和前面介绍的列组合、列偏移、列排序一样。需要清楚的是，当我们把列容器中的行容器的宽度设置为 100%时，其宽度和当前外部的列宽度是一样的。下面是案例展示，如图 11.11 所示。

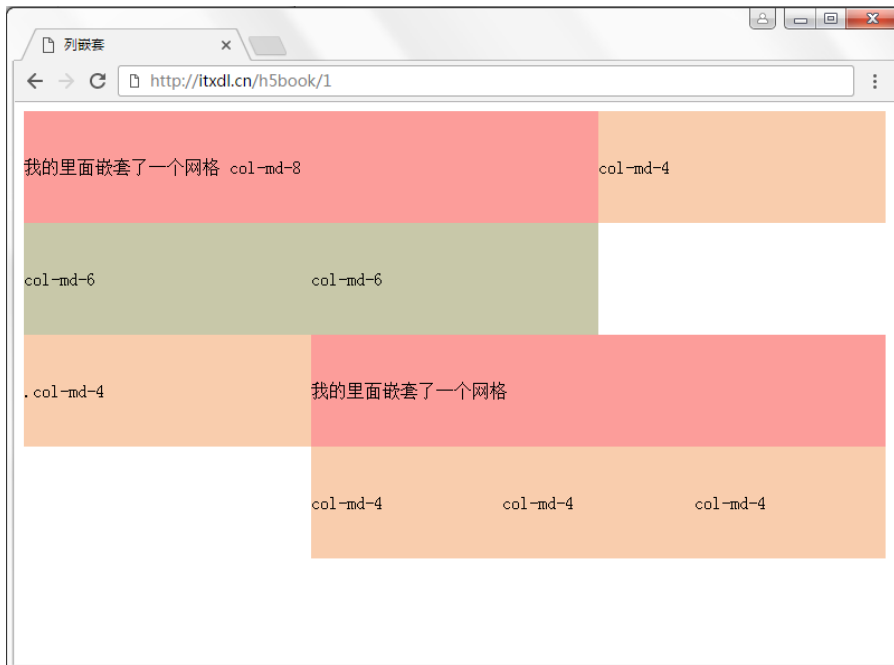


图 11.11 列嵌套

在这个案例中，第一行定义的是 8:4 的列宽，又在类名为“col-md-8”的列中定义了一个新的行(row)容器，在这个行容器中又重新定义了 6:6 的列宽。第二行中也是同样的嵌套思路，先在第二行里定义了 4:8 的列宽，在“col-md-8”的列中定义了新的行(row)容器，在这个行容器中划分了 4:4:4 的列宽。

源代码如下：



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>列嵌套</title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     .container{position: relative;width: 100%;}
8     .row{width:100%;float: left;}
9     .col-md-3,.col-md-4,.col-md-6,.col-md-8{
10    float: left;line-height:100px;}
11    .col-md-3{
12      width: 25%;
13      background: rgb(131,175,155);
14    }
15    .col-md-4{
16      width: 33.33333333%;
17      background: rgb(249,205,173);
18    }
19    .col-md-6{
20      width: 50%;
21      background: rgb(200,200,169);
22    }
23    .col-md-8{
24      width: 66.66666667%;
25      background: rgb(252,157,154);
26    }
27  </style>
28 </head>
29 <body>
30   <div class="container">
31     <div class="row">
32       <div class="col-md-8">
33         我的里面嵌套了一个网格 col-md-8
34         <div class="row">
35           <div class="col-md-6">col-md-6</div>
36           <div class="col-md-6">col-md-6</div>
37         </div>
38       </div>
39       <div class="col-md-4">col-md-4</div>
40     </div>
41     <div class="row">
42       <div class="col-md-4">.col-md-4</div>
43       <div class="col-md-8">
44         我的里面嵌套了一个网格
45         <div class="row">
46           <div class="col-md-4">col-md-4</div>
47           <div class="col-md-4">col-md-4</div>
48           <div class="col-md-4">col-md-4</div>
49         </div>
50       </div>
51     </div>
52   </div>
53 </body>
54 </html>
```

给出源代码后，读者就可以很清楚地看到这个 body 里的各个 DOM 节点，以及<div>标签是如何嵌套的。

3. 手机布局案例

在前面的基本用法讲解中，以中等屏幕（桌面显示器）为例进行了各种类的功能介绍。现在介绍手机端的布局案例，如图 11.12 和图 11.13 所示。

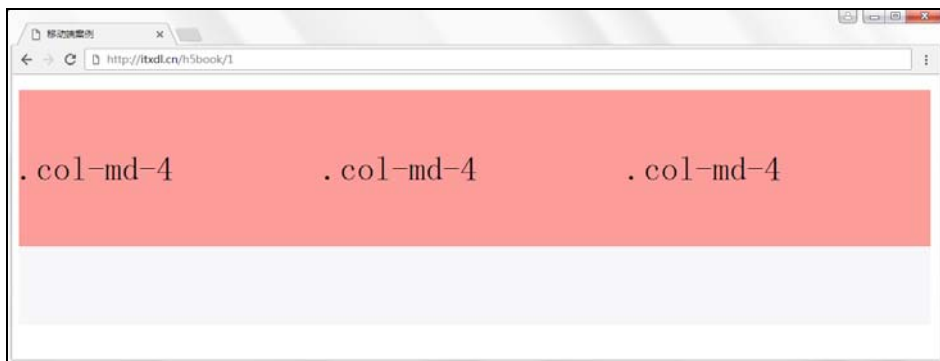


图 11.12 PC 版

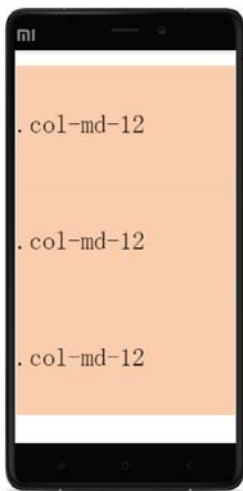


图 11.13 手机版

图 11.12 和图 11.13 显示的是两个不同屏幕的设备。可以看到，两者的布局也不一样，在图 11.12 中一行显示 3 列，每列占 4 个列的宽度；而在图 11.13 中，一行显示 1 列，每列占 12 个列的宽度。具体代码如下：

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>移动端案例</title>
5   <meta name="viewport" content="width=device-width, initial-scale=1.0,
6     maximum-scale=1.0, user-scalable=no">
7   <meta charset="utf-8">
   <style type="text/css">
```



```
8      *{list-style: none;margin: 0;padding: 0;}
9      .container{position: relative;height:300px;background:rgb(247,247,249);}
10     .row{width:100%;background: yellow;margin-top:20px;}
11     .col-md-3,.col-md-4,.col-md-6,.col-md-8,.col-md-12{
12     float: left;height: 200px;line-height:200px;}
13     /*确保所有列左浮动*/
14     .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6,
15     .col-md-7, .col-md-8, .col-md-9, .col-md-10, .col-md-11,
16     .col-md-12 {
17         float: left;
18         font-size: 44px;
19     }
20     /*定义每个列组合的宽度（使用百分比）*/
21     .col-md-12 {width: 100%;background: rgb(249,205,173);}
22     .col-md-11 {width: 91.66666667%;}
23     .col-md-10 {width: 83.33333333%;}
24     .col-md-9 {width: 75%;}
25     .col-md-8 {width: 66.66666667%;}
26     .col-md-7 {width: 58.33333333%;}
27     .col-md-6 {width: 50%;}
28     .col-md-5 {width: 41.66666667%;}
29     .col-md-4 {width: 33.33333333%;background: rgb(252,157,154);}
30     .col-md-3 {width: 25%;}
31     .col-md-2 {width: 16.66666667%;}
32     .col-md-1 {width: 8.33333333%;}
33     /*适配不同屏幕设配*/
34     @media (max-width: 768px) {
35         .container {
36             width: 750px;
37         }
38         .hidden-xs{
39             display: none;
40         }
41     }
42     @media (min-width: 992px) {
43         .container {
44             width: 1170px;
45         }
46         .hidden-md{
47             display: none;
48         }
49     }
50     </style>
51 </head>
52 <body>
53     <div class="container">
54         <div class="row">
55             <div class="col-md-4 hidden-xs ">.col-md-4</div>
56             <div class="col-md-4 hidden-xs">.col-md-4</div>
57             <div class="col-md-4 hidden-xs">.col-md-4</div>
58             <div class="col-md-12 hidden-md ">.col-md-12</div>
59             <div class="col-md-12 hidden-md ">.col-md-12</div>
60             <div class="col-md-12 hidden-md">.col-md-12</div>
61         </div>
62     </div>
63
64 </body>
65 </html>
```



可以看到，在源代码中，笔者只定义了一个行（row）容器，但为什么出现在两种设备上布局会不一样呢？这是因为笔者采用了@media 媒体查询，当监测到当前的屏幕宽度在 768px 以下时，就会加载类“hidden-xs”；当屏幕大于 992px 时，就会加载类“hidden-md”，而这两个类都是控制<div>标签的 display 属性并决定 div 模块是否在页面上显示出来。

当出现在移动设备上（屏幕 width<768px）时，就会使得第 55~57 行的 div 模块被隐藏，显示出来类名为“col-md-12”的 div 模块即移动端的布局。当不出现在移动设备上（屏幕 width>992px）时，第 58~60 行就会被隐藏，只显示出类名为“col-md-4”的 div 模块。

在以后的开发中，为了在不同的设备上显示不同的布局效果，就可以采用这种思路，先弄清楚想要在哪种设备上显示，哪种设备上不显示，然后将相应的“hidden-xs”、“hidden-md”、“hidden-sm”、“hidden-lg”添加到对应的<div>标签即可。

本章小结

通过这一章的学习，相信读者已经掌握了什么是栅格布局、栅格布局的原理以及如何使用栅格布局。重点掌握 Bootstrap 框架的栅格系统如何使用，包括列组合、列偏移、列排序及列嵌套。

本章习题

1. Bootstrap 框架中适用于桌面显示器的类前缀是下面的哪个选项？（ ）

- A. col-sm B. col-md C. col-xs D. col-lg

2. Bootstrap 中桌面显示器设备里想要实现向右偏移 3 列的类名是（ ）。

- A. col-md-offset-2 B. col-xs-offset-2
C. col-md-offset-3 D. Col-xs-offset-3

3. Bootstrap 中行中左右分别有类为“col-md-3”和“col-md-9”的两列布局，现在想把这两列布局的位置交换，右边显示“col-md-3”，左边显示“col-md-9”，那么需要在这两个类的基础上添加（ ）。

- A. “col-md-3 col-md-push-9”和“col-md-9 col-md-push-3”
B. “col-md-3 col-md-pull-3”和“col-md-9 col-md-push-9”
C. “col-md-3 col-md-push-9”和“col-md-9 col-md-pull-3”
D. “col-md-3 col-md-pull-9”和“col-md-9 col-md-pull-3”

4. Bootstrap 系统默认将 container 容器划分成（ ）列。

- A. 12 B. 24 C. 36 D. 48



5. 类“col-md-offset”是通过修改（ ）CSS 属性来实现效果的。

- A. padding B. margin C. left D. right



本章习题及其答案



本章资源包



本章扩展知识

第12章

Bootstrap 的快速入门



在前端开发中，开发人员会选择框架来帮助自己实现快速开发，而 Bootstrap 是目前一款很受欢迎的前端框架。Bootstrap 基于 HTML、CSS、JavaScript，简洁灵活，使得 Web 开发更加快捷。自 Bootstrap 3 起，它包含了贯穿于整个库的移动设备优先的样式，并且兼容各个主流浏览器。它的响应式 CSS，能够很好地自适应于台式机、平板电脑和手机。所以，这也是 Bootstrap 受到热捧的主要原因。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

12.1

Bootstrap 的内容概述与整体理解

通过前面的基础知识学习后，笔者将带领大家从整体上来学习 Bootstrap。首先，把 Bootstrap 的知识点划分为三大部分——全局 CSS 样式、组件和 JavaScript 插件。每部分都有各自的特点，其学习方式也有所不同。

【具体详细知识点，请参考 <http://v3.bootcss.com/中文文档>，笔者不会照搬照套手册，而是讲解如何快速、全方位地掌握 Bootstrap。】

接下来，笔者将对应 Bootstrap 的三大知识点，分别进行实例演示和讲解。



12.1.1 全局CSS样式

全局 CSS 样式是 Bootstrap 的基础，它统一了各个浏览器的样式风格，并且使开发人员从这枯燥而又耗时的 CSS 编写工作中解脱出来。

它的具体使用方式是通过属性声明来定义 HTML 标签的层叠样式，当然有部分 HTML 标签的默认层叠样式也被修改为统一风格了。

实例演示：

演示 Bootstrap 的全局 CSS 样式中的条纹表格样式。使用属性声明方式来定义其条纹表格——为<table>标签赋予 table 和 table-striped 两个属性即可。

实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>全局CSS样式演示</title>
8     <link href="css/bootstrap.min.css" rel="stylesheet">
9 </head>
10 <body>
11     <table class="table table-striped">
12         <thead>
13             <tr>
14                 <th>表格标题</th>
15                 <th>表格标题</th>
16                 <th>表格标题</th>
17             </tr>
18         </thead>
19         <tbody>
20             <tr>
21                 <td>表格单元格</td>
22                 <td>表格单元格</td>
23                 <td>表格单元格</td>
24             </tr>
25             <tr>
26                 <td>表格单元格</td>
27                 <td>表格单元格</td>
28                 <td>表格单元格</td>
29             </tr>
30             <tr>
31                 <td>表格单元格</td>
32                 <td>表格单元格</td>
33                 <td>表格单元格</td>
34             </tr>
35             <tr>
36                 <td>表格单元格</td>
37                 <td>表格单元格</td>
38                 <td>表格单元格</td>
39             </tr>
40         </tbody>
41     </table>
```

```
42 <script src="js/jquery-2.2.4.min.js"></script>
43 <script src="js/bootstrap.min.js"></script>
44 </body>
45 </html>
```

运行结果如图 12.1 所示。



图 12.1 全局 CSS 样式演示

其中的原理是，Bootstrap 通过动态 CSS——Less/Sass 语言来进行预编译，生成一整套 CSS 全局样式。因此，读者可以通过修改 Less 或 Sass 中的变量，来修改 Bootstrap 的默认样式。当然，如果读者对预编译 CSS 语言运用得足够熟练，也可以自定义属于自己专有风格的全局 CSS 样式——无论是添加新的组件或删除组件或修改默认样式(大型网站一般都会使用 Sass 来定制一套属于本网站的样式库和组件库，但本书主要讲解 Bootstrap 默认的全局样式，自定义的样式为辅)。

全局 CSS 样式在 Bootstrap 使用中是最基本的单元，也是最小单元，这是非常重要的知识点，特别是在对某些细小部分进行微调时，它的作用就体现出来了。

12.1.2 组件

组件是 Bootstrap 的核心内容，也是运用最频繁的部分。之所以 Bootstrap 受到众人追捧，是因为其组件的通用性极强，它把常见的页面布局全部封装到一个个的小组件中。当开发者进行页面布局的时候，完全可以使用搭积木的思想，一个个积木直接搭建而成，而无须再去关注那些枯燥乏味而又烦琐的 CSS 了。

实例演示：

演示 Bootstrap 的组件中的导航条。使用属性声明方式来布局两种不同样式的导航条。



实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>bootstrap的组件</title>
8   <link href="css/bootstrap.min.css" rel="stylesheet">
9 </head>
10 <body>
11   <ul class="nav nav-tabs">
12     <li class="active"><a href="#">Home</a></li>
13     <li><a href="#">CSS3</a></li>
14     <li><a href="#">Sass</a></li>
15     <li><a href="#">jQuery</a></li>
16     <li class="disabled"><a href="#">Responsive</a></li>
17   </ul>
18   <br>
19   <ul class="nav nav-pills">
20     <li class="active"><a href="#">首页</a></li>
21     <li class="dropdown">
22       <a href="#" class="dropdown-toggle" data-toggle="dropdown">教程
23       <span class="caret"></span></a>
24       <ul class="dropdown-menu">
25         <li><a href="#">CSS3</a></li>
26         <li><a href="#">Sass</a></li>
27         <li><a href="#">jQuery</a></li>
28         <li><a href="#">Responsive</a></li>
29       </ul>
30     </li>
31     <li><a href="#">关于我们</a></li>
32   </ul>
33   <script src="js/jquery-2.2.4.min.js"></script>
34   <script src="js/bootstrap.min.js"></script>
35 </body>
</html>
```

运行结果如图 12.2 所示。

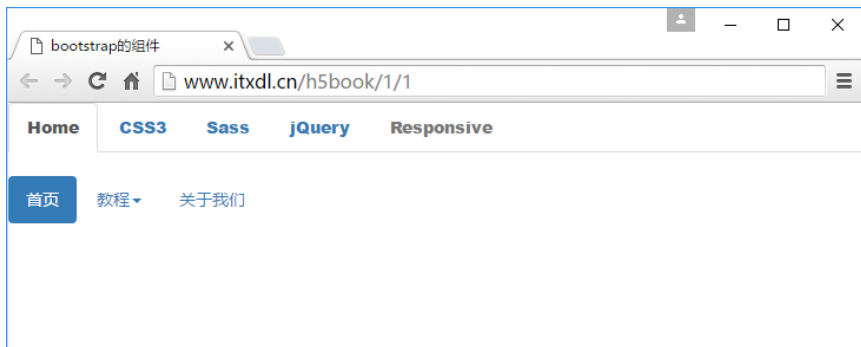


图 12.2 组件演示

当然，读者也可以在 Sass 文件中添加其他组件或修改、删除系统自带组件。

12.1.3 JavaScript插件

JavaScript 插件在 Bootstrap 中也是主要通过 HTML 的属性定义式来实现 HTML 页面的动画效果，它把所有常见的 js 特效和一些与自身相关的组件特效，全部封装到 Bootstrap 内部的 js 文件中，并命名为 bootstrap.min.js。

bootstrap.min.js 形成的动画效果完全依赖于 jQuery。而 Bootstrap 自身的 JavaScript 插件并不是很多，因为如今 Web 页面中的特效五花八门，每个页面都会有一套自己的风格。

实例演示：

演示 Bootstrap 的 JavaScript 插件中的模态框。

实例代码：

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>JavaScript插件的模态框</title>
8     <link href="css/bootstrap.min.css" rel="stylesheet">
9 </head>
10 <body>
11     <button class="btn btn-primary" data-toggle="modal" data-target=
12     "#myModal-data" type="button">通过data-target触发</button>
13     <!-- 模态弹出窗内容 -->
14     <div class="modal" id="myModal-data" tabindex="-1" role="dialog"
15     aria-labelledby="mySmallModalLabel" aria-hidden="true">
16         <div class="modal-dialog">
17             <div class="modal-content">
18                 <div class="modal-header">
19                     <button type="button" class="close" data-dismiss="modal"
20                     ><span aria-hidden="true">&times;</span><span class=
21                     "sr-only">Close</span></button>
22                     <h4 class="modal-title">模态弹出窗标题</h4>
23                 </div>
24                 <div class="modal-body">
25                     <p>模态弹出窗主体内容</p>
26                 </div>
27                 <div class="modal-footer">
28                     <button type="button" class="btn btn-default"
29                     data-dismiss="modal">关闭</button>
30                     <button type="button" class="btn btn-primary">保存
31                     </button>
32                 </div>
33             </div>
34         </div>
35     </div>
36     <script src="js/jquery-2.2.4.min.js"></script>
37     <script src="js/bootstrap.min.js"></script>
38 </body>
39 </html>

```

运行结果如图 12.3 所示。

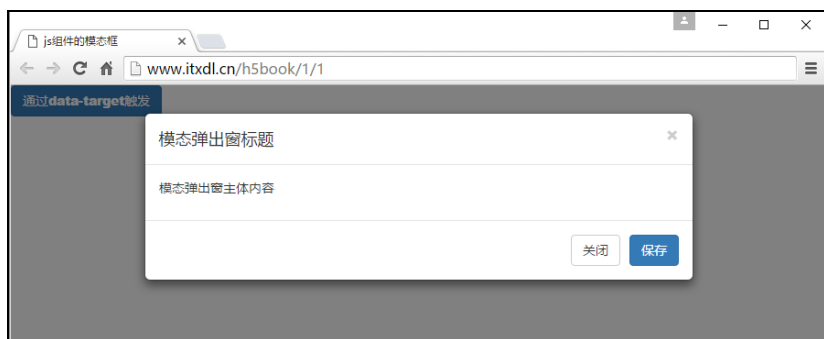


图 12.3 JavaScript 插件演示

简单学习完上述三大知识点后,笔者再次强调,Bootstrap 的详细知识点请参考官网手册,本书则主要从整体入手,并做一个实战项目,来贯通 Bootstrap 的整体知识点。

【学习本章时,希望读者能对照 Bootstrap 手册进行同步学习,完成本章学习之后,至少对 Bootstrap 手册有一个感性的认知。】

12.2 Bootstrap搭建环境

搭建 Bootstrap 环境首先要下载 Bootstrap 资源包,这里笔者下载的资源包为 Sass 源码版本的 Bootstrap (如果是手写 CSS 样式,则可以考虑下载生成环境下的 Bootstrap)。一来巩固 Sass 知识,二来学习如何使用 Sass 来自定义自己风格的模块。

第一步,首先使用 sass 命令编译出 Bootstrap 默认的 CSS 文件。本书下载的 Sass 包名为 bootstrap-3.3.0-sass.tar.gz。在终端命令行中,执行以下命令:

```
sass --watch ./assets/stylesheets/_bootstrap.scss:mybootstrap.css
```

这样就可以生成一套可用的 Bootstrap 层叠样式文件——mybootstrap.css,然后引入到基本模板中,此时 Bootstrap 环境就算搭建完成了(如果是下载生成环境包,则直接引入 bootstrap.css 即可)。

基本模板代码:

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <!-- 上述3个meta标签*必须*放在最前面,任何其他内容都*必须*跟随其后! -->
8     <title>Bootstrap 101 Template</title>
9
10    <!-- Bootstrap -->
11    <link href="css/mybootstrap.min.css" rel="stylesheet">
```

```

12
13 <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media queries -->
14 <!-- WARNING: Respond.js doesn't work if you view the page via file://
-->
15 <!--[if lt IE 9]>
16 | <script
| src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"></script>
17 | <script
| src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>
18 <![endif]-->
19 </head>
20 <body>
21 <h1>你好，世界！</h1>
22
23 <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
24 <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></script>
25 <!-- Include all compiled plugins (below), or include individual files
as needed -->
26 <script src="js/bootstrap.min.js"></script>
27 </body>
28 </html>

```

对此模板代码的理解如下。

第 6 行代码：为了让网页更好地去适配或响应不同分辨率的移动端（让其虚拟像素等于移动设备的实际宽度，并且设置虚拟像素和物理像素比为 1）。

第 11 行代码：引入 Bootstrap 的全局 CSS 样式，包括其组件内的样式。我们引入的是 mybootstrap.css（如果是下载生成环境包，则直接引入 bootstrap.css）。

第 16~17 行代码：如果是 IE 低端浏览器，则不支持 HTML5 和 CSS3，这时可以打开这两行代码，让低端 IE 浏览器也可以正常渲染 Bootstrap。

第 24 行代码：为 jQuery 文件（个人建议开发使用本文文件，而不使用 CDN 文件）。

第 26 行代码：引入 Bootstrap 的 JavaScript 插件文件。

执行上述基本模板代码，整个 Bootstrap 即可开始运行。

12.3 Bootstrap 全局 CSS 样式

Bootstrap 的全局 CSS 样式属于其框架基本知识点，它是整个框架的基石，非常重要。其所有的知识点必须灵活掌握，在此笔者将对 Bootstrap 的全局 CSS 样式进行总结，如图 12.4 所示。

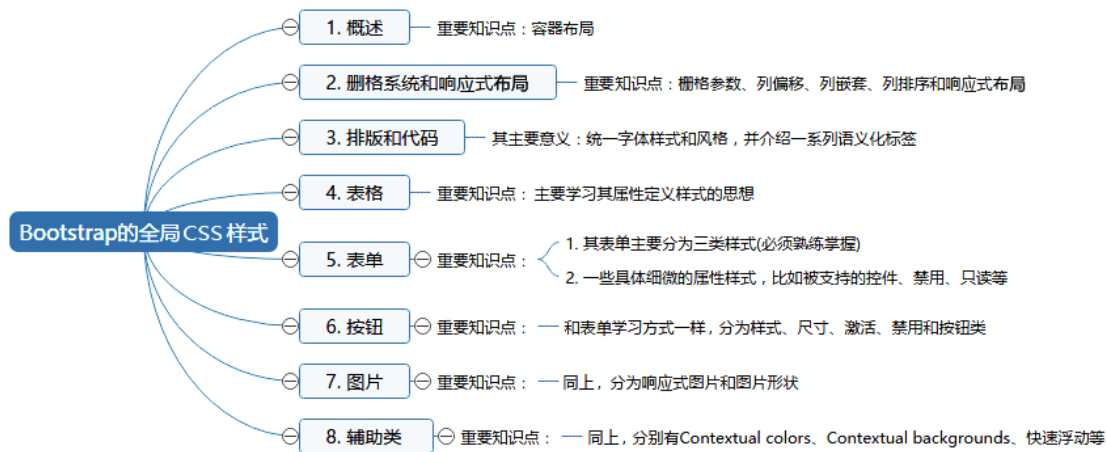


图 12.4 Bootstrap 的全局 CSS 样式知识点概括

参考上述结构图，对照官方手册进行详细学习，并做出相应的实例加以巩固。下面笔者将带领大家学习上述部分知识点。

12.3.1 全局CSS样式的栅格系统和响应式布局

所谓栅格系统，就好比日常的衣柜。如果一个衣柜没有隔层，所有的衣服全部堆放在一起，就算堆放得整齐有序，也会给人一种拥挤和凌乱的感觉。而栅格系统的工作，就好比这个衣柜的隔层，衣服全部按照类型分开搁放，给人一种井然有序的感觉，并且对日后维护和管理起到非常大的作用。

所以在 Bootstrap 手册中，读者需要着重学习栅格参数、列偏移、列嵌套和列排序。

而 Bootstrap 的栅格系统应该具有另一个意义，即响应式布局。这也是项目中运用得最多的。

实例描述：

附加栅格参数、列偏移和列排序等知识点，完成一个响应式栅格布局。

实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>响应式布局</title>
8     <link href="css/bootstrap.min.css" rel="stylesheet">
9 </head>
10 <body>
11 <div class="container-fluid">
12     <div class="row"> <!-- 响应式布局 -->
13         <div class="col-xs-6 col-sm-4 col-md-3">小屏2列中屏幕3大屏4列</div>
```

```

14     <div class="col-xs-6 col-sm-4 col-md-3">小屏2列中屏幕3大屏4列</div>
15     <div class="hidden-xs col-sm-4 col-md-3">小屏2列中屏幕3大屏4列</div>
16     <div class="hidden-xs hidden-sm col-md-3">小屏2列中屏幕3大屏4列
17     </div>
18     <div class="row">    <!-- 列的排序 -->
19     <div class="col-xs-6 col-xs-push-6 col-sm-4 col-sm-push-4 col-md-3
20     col-md-push-3">小屏2列中屏幕3大屏4列</div>
21     <div class="col-xs-6 col-xs-pull-7 col-sm-4 col-sm-pull-5 col-md-3
22     col-md-pull-4">小屏2列中屏幕3大屏4列</div>
23     </div>
24     <div class="row">    <!-- 列的偏移量 -->
25     <div class="col-xs-6 col-xs-offset-6 col-sm-4 col-sm-offset-4
26     col-md-3 col-md-offset-3">小屏2列中屏幕3大屏4列</div>
27     <div class="hidden-xs col-sm-4 col-md-3">小屏2列中屏幕3大屏4列</div>
28     <div class="hidden-xs hidden-sm col-md-3">小屏2列中屏幕3大屏4列
29     </div>
30     </div>
31 </div>
<script src="js/jquery-2.2.4.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>

```

运行结果如图 12.5~图 12.7 所示。

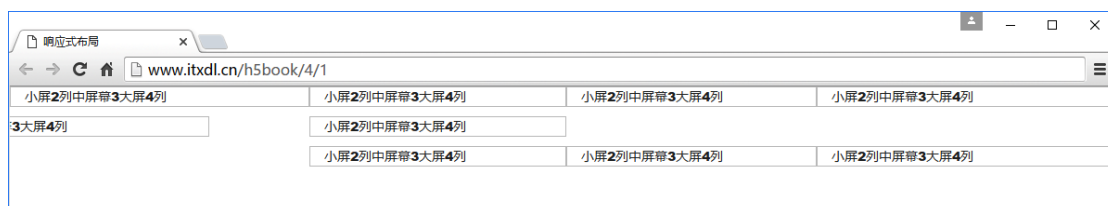


图 12.5 响应式布局大屏运行结果

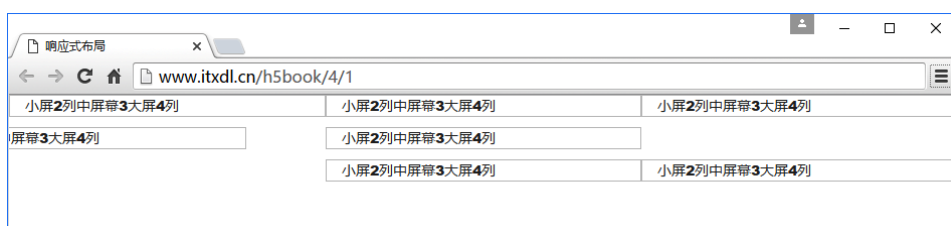


图 12.6 响应式布局中屏运行结果

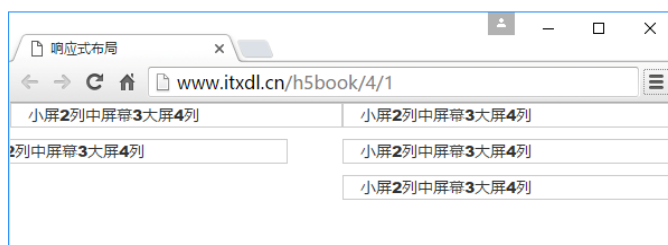


图 12.7 响应式布局小屏运行结果



可以看到, Bootstrap 通过.col-xs-*、.col-sm-*、.col-md-*和.col-lg-* 4 种类名进行响应式适配, 而每种适配使用 12 列栅格进行页面布局。对应的还有.visible-xs-*、.visible-ms-*、.visible-md-*和.visible-lg-*, 以及.hidden-xs、.hidden-ms、.hidden-md 和 .hidden-lg 8 种类名来辅助响应式的适配, 针对不同屏幕尺寸隐藏或显示页面内容。

12.3.2 全局CSS样式的表单

全局 CSS 样式的表单, 使用属性来定义元素的层叠样式。由于 HTML5 的表单标签很多, 使得在 Bootstrap 中也必须一一对应。

在表单中主要分为基本样式表单、内联样式表单和水平排列表单 3 种样式, 如表 12.1 所示。

表 12.1 Bootstrap 表单主要样式

表单样式	class 属性	描 述
基本样式表单	form-control	为 Bootstrap 对表单组件样式控制的属性 (组件宽度为 100%)
	form-group	可以让每个组件之间隔离, 以此来获取最好的样式
内联样式表单	form-inline	可以让表单的内容左对齐, 形成行内样式, 但当视口 (viewport) 小于 768px 时, 表单将会折叠
	form-horizontal	使用行内样式时, <label>标签必须使用此属性来支持屏幕阅读器
	input-group-addon 和 input-group	两个属性组合使用, 使得表单组件和文字 title 被一个边框包裹起来
水平排列表单	form-horizontal	利用 Bootstrap 的栅格特性, 完成水平的排列 [不同的是, row 的特性被 form-grop 代替, label 也可以使用 column 的属性]

读者可以尝试参考 Bootstrap 手册完成上述 3 种表单。其中, 里面还涉及一些具体细微的属性样式, 如被支持的控件、静态控件、焦点状态、禁用状态、只读状态、校验状态、控件大小和辅助文本等。目前, 这些只做了解, 在下一章将会进行详细讲解。

12.4 Bootstrap组件

Bootstrap 的组件是最核心的内容, 也是使用最多的部分。它集成了非常多的通用组件, 一般的页面布局足够使用。页面开始会使用 Bootstrap 的栅格系统把页面分隔为一个个小区块, 然后使用 Bootstrap 的组件进行填充装饰, 即可轻松完成整张页面的布局。下面是 Bootstrap 组件的概述图, 如图 12.8 所示。

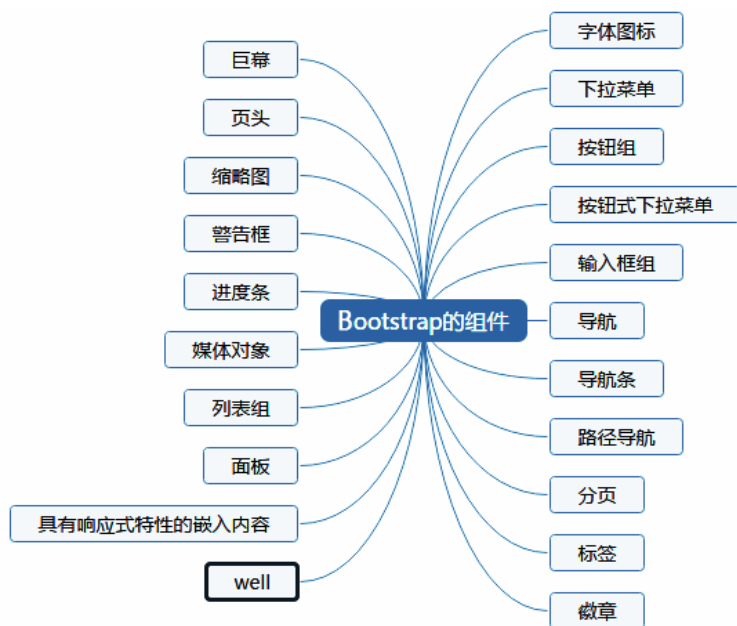


图 12.8 Bootstrap 所有组件的示意图

由上可以看出，Bootstrap 的组件多达 21 个，而且里面的组件都可以灵活搭配使用。下面笔者将演示一个常见的功能——面板。

实例代码：

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>组件—面板</title>
8   <link href="css/bootstrap.min.css" rel="stylesheet">
9   <style>
10    body{padding:20px;}
11  </style>
12 </head>
13 <body>
14 <div class="panel panel-default">
15   <div class="panel-heading">兄弟连教育</div>
16   <div class="panel-body">
17     <p>让学习成为一种习惯</p>
18   </div>
19 </div>
20 <div class="panel panel-default">
21   <div class="panel-heading">猿代码</div>
22   <div class="panel-body">
23     <p>兄弟连教育平台</p>
24   </div>
25 </div>
26 <script src="js/jquery-2.2.4.min.js"></script>
27 <script src="js/bootstrap.min.js"></script>

```




```
28 </body>
29 </html>
```

运行结果如图 12.9 所示。

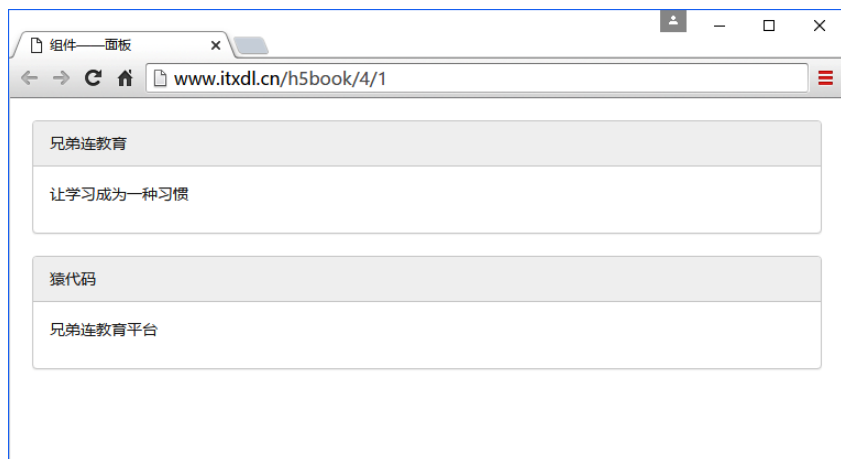


图 12.9 组件中的面板

其余组件的使用也是同理，另外配合好全局 CSS 样式，能更加灵活地运用其组件。而 Bootstrap 组件的实际运用，笔者放在下一章讲解。

12.5 Bootstrap的JavaScript插件

Bootstrap 的 JavaScript 插件提供了常见的几种特效，但是它不提供第三方 JavaScript 工具库的支持，当需要引入其他 JavaScript 库文件时，需要注意解决其冲突问题。

实例描述：

继续 Bootstrap 面板操作，完成面板可收缩、可关闭的特效。

实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>JavaScript实例—面板</title>
8     <link href="css/bootstrap.min.css" rel="stylesheet">
9     <style>
10         body{padding:20px;}
11         .alert{padding:0px;}
12     </style>
13 </head>
```

```

14 <body>
15 <div class="panel-group" id="accordion">
16   <div class="panel panel-default alert" role="alert">
17     <div class="panel-heading">
18       <h4 class="panel-title">
19         <a data-toggle="collapse" data-parent="#accordion" href=
20           "#collapseOne">兄弟连教育</a>
21         <button class="close" data-dismiss="alert" type="button" >
22           &times;</button>
23       </h4>
24     </div>
25     <div id="collapseOne" class="panel-collapse collapse in">
26       <div class="panel-body">让学习成为一种习惯</div>
27     </div>
28   <div class="panel panel-default alert" role="alert">
29     <div class="panel-heading">
30       <h4 class="panel-title">
31         <a data-toggle="collapse" data-parent="#accordion" href=
32           "#collapseTwo">猿代码</a>
33         <button class="close" data-dismiss="alert" type="button" >
34           &times;</button>
35       </h4>
36     </div>
37     <div id="collapseTwo" class="panel-collapse collapse">
38       <div class="panel-body">兄弟连教育平台</div>
39     </div>
40   <div class="panel panel-default alert" role="alert">
41     <div class="panel-heading">
42       <h4 class="panel-title">
43         <a data-toggle="collapse" data-parent="#accordion" href=
44           "#collapseThree">兄弟连教育</a>
45         <button class="close" data-dismiss="alert" type="button" >
46           &times;</button>
47       </h4>
48     </div>
49     <div id="collapseThree" class="panel-collapse collapse">
50       <div class="panel-body">让学习成为一种习惯</div>
51     </div>
52   </div>
53 </div>
54 <script src="js/jquery-2.2.4.min.js"></script>
55 <script src="js/bootstrap.min.js"></script>
56 </body>
57 </html>

```

运行结果如图 12.10 所示。

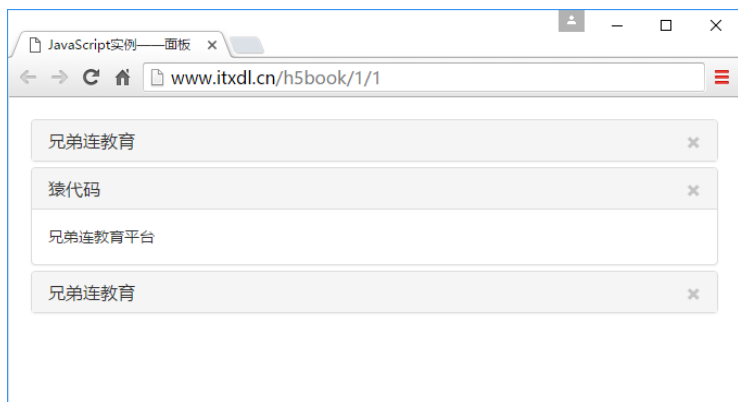


图 12.10 JavaScript 插件中的面板

本章小结

本章主要是快速学习 Bootstrap。首先分别概述 Bootstrap 的三大知识点——全局 CSS 样式、组件和 JavaScript 插件，然后分别进一步分析每个知识点并给出相应的实例。

下一章为一个实战章节，笔者将带领大家活学活用 Bootstrap，掌握其中的精髓。

本章习题

1. 有一个元素，需要在 PC 端显示而在手机端隐藏，需要使用的方法是（ ）。
A. visible-xs-8 hidden-md B. visible-md-8 hidden-xs
C. visible-md-8 hidden-sm D. visible-sm-8 hidden-md
2. 有一个元素，需要在打印时隐藏，使用的类是（ ）。
A. visible-print-block B. visible-print-inline
C. hidden-print D. print-hidden
3. 根据栅格系统的标准用法，错误的是（ ）
A. <div class="container">
 <div class="row">
B. <div class="row">
 <div class="col-md-1">
C. <div class="row">
 <div class="container">
D. <div class="col-md-1">
 <div class="row">
4. 下列哪些是正确的辅助类？【多选】（ ）
A. text-muted B. text-success C. text-info D. text-warning

5. Bootstrap 提供了哪几套代码? 【多选】()
- A. 生成环境源码 B. Less 源码 C. Sass 源码 D. Iaas 源码
6. 对 Bootstrap 的描述正确的是 ()。
- A. Bootstrap 是一个前端文件库
- B. Bootstrap 是一个框架
- C. Bootstrap 对所有的全局样式进行了统一规范
- D. Bootstrap 可以适配 3 种屏幕的响应
7. Bootstrap 默认使用的栅格系统是多少栏的? ()
- A. 6 栏 B. 12 栏 C. 16 栏 D. 32 栏
8. 下面对 Bootstrap 的基本环境搭建叙述正确的是 ()。
- A. Bootstrap 默认支持 IE 8 浏览器
- B. Bootstrap 使用的 JavaScript 库文件是 jQuery
- C. 在移动端中 Bootstrap 会自动加载 zepto.js 来替代 jQuery
- D. Bootstrap 默认设置了其虚拟像素和物理像素比为 1
9. Bootstrap 对以下哪些内容进行封装? 【多选】()
- A. 对全局 CSS 样式进行封装 B. 对组件进行封装
- C. 对 Ajax 请求进行封装 D. 对 JavaScript 插件进行封装
10. 简述题: 请简单描述一下 Bootstrap 中全局 CSS 样式、组件样式和 JavaScript 插件是通过何种方式渲染页面的?



本章习题及其答案



本章资源包



本章扩展知识

第13章

Bootstrap 的实战



上一章快速了解了 Bootstrap 的知识结构，并学会了简单地运用 Bootstrap。而本章笔者将带领大家用 Bootstrap 来做一个完整的实战项目（猿代码直播平台首页），主要讲解 Bootstrap 默认的全局 CSS 样式、组件和 JavaScript 插件，辅助讲解如何使用 Sass 来修改默认样式或添加新的组件。



本章二维码

本章二维码里面包括：

- (1) 本章的学习视频；
- (2) 本章所有实例演示结果；
- (3) 本章习题及其答案；
- (4) 本章资源包（包括本章所有代码）下载；
- (5) 本章的扩展知识。

13.1 实战概述

本章主体内容是做一个完整的实战页面。首先，笔者会根据直播平台首页的实际需求，划分为一个个具体的功能模块。其次，把功能模块拆分为一个个 Bootstrap 知识点，分别讲解具体的知识点，然后完成实战项目的内容。当然，每个页面都有独特的风格，笔者根据实际情况使用 Sass 来自定义一些全局 CSS 样式和组件。

13.2 实战需求

首先，看看直播平台的实际需求，大致分为 5 个具体模块——顶部的工具栏、页面的导

航条、banner 区、推荐位和脚部。而每个具体模块又包含以下内容：

- (1) 顶部的工具栏中包含网站 Logo、个人操作导航、搜索和个人状态栏目。
- (2) 页面的导航条为整个网站的内容模块的导航。
- (3) banner（横幅）区为一个轮播图效果。
- (4) 推荐位可以按照不同风格写多套样式，里面包括推荐位导航和推荐位内容。
- (5) 脚部为本网站或公司的基本信息介绍。

实战的预想结果如图 13.1～图 13.3 所示。



图 13.1 实战首页展示图——大屏样式



上述页面样式为大屏的显示样式，如 PC 电脑屏幕等，其在 Bootstrap 中默认的严格标准是屏幕横屏分辨率大于 1200px 时，都算大屏幕样式。其中，有部分 PC 电脑屏幕偏小，但屏幕横屏大于 992px、小于 1200px，这在 Bootstrap 中默认归为中等屏幕。

中等屏幕显示的样式和大屏幕的显示样式几乎相同，在此就不再展示了。



图 13.2 实战首页展示图——小屏样式

上述页面样式为小屏的显示样式，一般指平板电脑等大屏移动设备。其在 Bootstrap 中默认的严格标准是屏幕横屏分辨率大于 768px、小于 992px，即归为小屏幕样式。



图 13.3 实战首页展示图——超小屏幕样式

上述页面样式为超小屏幕的显示样式，一般指手机等小屏移动设备。其在 Bootstrap 中默认的严格标准是屏幕横屏分辨率小于 768px，即归为超小屏幕样式。

接下来，正式进入实战项目。

13.3 实战准备

首先来看实战项目需要的目录结构，如下所示：

```

1 | 
2 | |---css-----CSS文件夹
3 | |   |--bootstrap-----官方Bootstrap的Sass库目录
4 | |   |   `-----略
5 | |   |--mybootstrap-----自定义的Sass库目录
6 | |       |--mixins-----Sass库中的混合宏目录
7 | |           |--_border.scss-----边框的混合宏文件
8 | |           |--_border.scss-----边框的全局CSS样式
9 | |           |--_color.scss-----色彩的全局CSS样式
10 | |           |--_font.scss-----字体的全局CSS样式
11 | |           |--_mixins.scss-----引入混合宏的入口文件
12 | |           |--_mystyle.scss-----自定义组件
13 | |           |--_paddingOrMargin.scss-----边距的全局CSS样式
14 | |           |--_variables.scss-----变量定义
15 | |       |--_bootstrap.scss-----Sass的入口文件
16 | |       |--mybootstrap.css-----Sass编译后生成的CSS文件
17 | |---fonts-----字体文件
18 | |   `-----略
  
```




```
19 | ----images-----图片目录
20 | ----js-----js文件目录
21 | | ----bootstrap.js
22 | | `----jquery-2.2.4.min.js
23 | ----index.html-----项目文件
```

笔者使用 bootstrap-3.3.0-sass.tar.gz 完成此项目。从上面可以看出，css/bootstrap 目录是官方 Bootstrap 的 Sass 库目录（在 tar 包中位置为 bootstrap-3.3.0-sass\bootstrap-sass-3.3.0\assets\stylesheets），与此同级目录下有一个 _bootstrap.scss 文件，它是 Sass 库编译的入口文件。而目录结构中 css 目录下的其他 Sass 文件都是手动建立文件（指 ./css/mybootstrap 目录下的所有文件）。mybootstrap.css 文件则是通过 Sass 编译出来的 CSS 文件，后面笔者将通过 _bootstrap.scss 文件把 Bootstrap 默认的 Sass 库和自定义的 Sass 库关联起来，然后生成此文件。

同理，把 tar 包中的 font 字体库也复制一份到上述的 font 目录下（目录结构中字体库目录结构省略）。注意一点，Bootstrap 的生产包和 Sass 包里面的字体库目录结构有稍许不同。

接着，新建的 images 目录用来存放项目的图片，新建的 js 目录用来存放项目的 JavaScript 文件。

最后，新建 index.html 文件则为项目文件。

13.3.1 Sass配置

在前面的内容中，笔者说过用户完全可以修改 Bootstrap 默认的样式。在本项目中，笔者以一处为示例：将栅格系统的内补白和外补白全部修改为 0px，其修改代码为 bootstrap/_variables.scss 中的第 325 行：

```
$grid-gutter-width: 0px !default; //本项目仅修改此处，其他不变
```

【本章的主要目的是学习如何运用 Bootstrap，但又要做出网站独有的风格，为了方便教学，笔者自定义了一套 Sass 库文件，采用样式覆盖的原则来进行渲染页面层叠样式。】

本项目使用 Bootstrap 默认的 Sass 库文件的同时，笔者也自定义了一套私有的 Sass 库——mybootstrap 目录。它包含七个文件和一个目录，同官方的 Bootstrap 架构一样，并且里面内容包括自定义的全局 CSS 样式、自定义的组件和自定义的 js 样式。同理，mybootstrap/mixins 目录全部是上一级目录文件调用的混合宏。

下面笔者通过入口文件 _bootstrap.scss 文件，把自定义的 Sass 文件全部加入到项目中，如下所示。

加载自定义 bootstrap 文件：

```
52 //myBootstrap CSS
53 @import "myBootstrap/variables";
54 @import "myBootstrap/mixins";
```

```

55 @import "myBootstrap/font";
56 @import "myBootstrap/border";
57 @import "myBootstrap/color";
58 @import "myBootstrap/paddingOrMargin";
59 @import "myBootstrap/mystyle";

```

接下来, 通过 mybootstrap.scss/mixins.scss 把所有自定义的混合宏文件全部导入到项目中, 如下所示。

加载自定义的混合宏文件:

```

1
2 @import "mixins/border";

```

最后, 直接编译项目的 Scss 库文件, 动态生成 CSS 文件, 此刻项目的 Sass 配置基本完成。命令如下:

```
sass --watch ./css/_bootstrap.scss:./css/mybootstrap.css
```

13.3.2 HTML的基本模块

HTML 页面布局的第一步是为页面划分区块。每个区块都是一个独立的模块, 互不影响, 从而可以进行拼凑来形成一个完整的页面。而 Bootstrap 的栅格系统和响应式布局就可以实现每个模块的划分, 并支持不同视口的宽度而形成的响应式布局效果。

实例代码:

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title></title>
8     <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11 <!-- 顶部工具栏 -->
12 <header id="head">顶部工具栏</header>
13 <!-- 页面导航条-->
14 <nav id="pageNav">页面导航条</nav>
15 <!-- banner -->
16 <div id="carousel">banner</div>
17 <!-- 内容栏目 -->
18 <div id="pageContainer">内部栏目</div>
19 <!-- 脚部 -->
20 <footer id="footer">脚部</footer>
21 <script src="js/jquery-2.2.4.min.js"></script>
22 <script src="js/bootstrap.js"></script>
23 </body>
24 </html>

```

到此, HTML 的基本模块已经部署完成了, 下面笔者将按照模块划分的顺序依次实现。



13.4 顶部工具栏

顶部工具栏一共分为四部分内容：Logo、个人操作导航、搜索和个人状态栏目。在 PC 端布局中，可以把这四部分内容放在一行进行显示，但在移动端布局中就必须另做处理了，四部分内容完全放置到一行会显得非常臃肿，甚至完全摆放不下。

因此，笔者将顶部工具栏进一步细分为移动端区域和 PC 端区域，如下所示：

```
11 <!-- 顶部工具栏 -->
12 <header id="head">
13     <header class="container">
14         <!-- 移动端 -->
15         <div class="hidden-sm hidden-md hidden-lg">移动端</div>
16         <!-- PC 端 -->
17         <div class="hidden-xs">pc端</div>
18     </header>
19 </header>
```

从上述代码可以发现：

(1) 页面#head 元素节点为顶部工具栏容器，它是一个流体容器，可以通过此节点来设置整个顶部工具栏的背景颜色或边框。

(2) 页面#head.container 元素节点为固定容器，里面主要存放顶部工具栏实体内容，如上述的移动端实体内容和 PC 端实体内容。

(3) PC 端和移动端区块的划分是根据 Bootstrap 提供的全局 CSS 样式中的响应式工具来完成的，通过不同设备的横屏分辨率来完成不同窗口页面的适配。

现在，笔者按照实际需求，把 PC 端和移动端的基本内容做了不同的规划和设计。

顶部工具栏 PC 端基本设计：使用栅格系统把其分为四栏，其中 Logo 部分和个人状态栏目直接显示，而个人操作导航则使用 Bootstrap 的导航组件，搜索则使用 Bootstrap 的输入框组件。

顶部工具栏移动端基本设计：由于移动端（比如手机）屏幕宽度有限，只能把 Logo 省去，换成醒目的标题；而把个人操作导航、搜索框和个人状态栏目等使用 Bootstrap 的下拉菜单进行布局显示。

现在，读者首先学习一下 Bootstrap 系统默认的全局 CSS 样式和组件，然后再完成实战项目内容。

13.4.1 Bootstrap的字体图标组件

字体图标是非常常用的一类组件，当然 Bootstrap 的默认库也有一套自己的字体图标组

件。其字体图标文件默认存放在../fonts/bootstrap 目录内，使用的时候需要一个基类和对应的图标类，这样性能会更好，如下所示。

实例代码：

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>字体图标</title>
8   <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11 <h1>
12   <span class="glyphicon glyphicon-asterisk"></span>
13   <span class="glyphicon glyphicon-plus"></span>
14   <span class="glyphicon glyphicon-euro"></span>
15 </h1>
16 <script src="js/jquery-2.2.4.min.js"></script>
17 <script src="js/bootstrap.js"></script>
18 </body>
19 </html>

```

运行结果如图 13.4 所示。

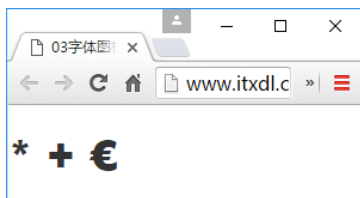


图 13.4 字体图标的运行结果

当然，读者可以选择更加丰富的字体图标组件引入到本项目中来，其运用大同小异。

13.4.2 Bootstrap 的下拉菜单组件

下拉菜单组件为 Bootstrap 的一个基础组件，因为在 Bootstrap 的某些高级组件中是包含此组件的运用的。在下拉菜单组件中常见的知识点有：对齐、标题、分割线和禁用的菜单项。其具体运用如下所示。

实例代码：

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">

```



```
7 <title></title>
8 <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <style>
11     body{padding:50px;}
12     .dropdown{float:left;}
13 </style>
14 <body>
15 <div class="dropdown">
16     <button class="btn btn-default dropdown-toggle" type="button" id=
17         "dropdownMenu1" data-toggle="dropdown">
18         左对齐下拉菜单
19     </button>
20     <ul class="dropdown-menu dropdown-menu-left" role="menu" aria-labelledby=
21         "dropdownMenu1">
22         <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
23             选项一</a></li>
24     </ul>
25 </div>
26 <div class="dropdown">
27     <button class="btn btn-default dropdown-toggle" type="button" id=
28         "dropdownMenu2" data-toggle="dropdown">
29         右对齐下拉菜单
30     </button>
31     <ul class="dropdown-menu dropdown-menu-right" role="menu" aria-labelledby=
32         "dropdownMenu2">
33         <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
34             选项一</a></li>
35         <li role="presentation" class="divider"></li>
36         <li role="presentation" class="dropdown-header">选项标题</li>
37         <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
38             选项二</a></li>
39         <li role="presentation" class="disabled"><a role="menuitem" tabindex=
40             "-1" href="#">选项三（禁选项）</a></li>
41     </ul>
42 </div>
43 <script src="js/jquery-2.2.4.min.js"></script>
44 <script src="js/bootstrap.js"></script>
45 </body>
46 </html>
```

运行结果如图 13.5 所示。

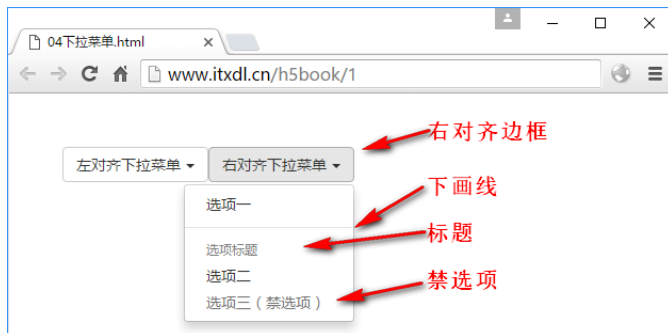


图 13.5 Bootstrap 的下拉菜单组件运行结果

从上述运行结果可以发现, Bootstrap 可以非常轻松地完成下拉菜单的各种样式, 比如右对齐、下划线、标题和禁用选项等, 仅需要一个类或一行代码即可完成。

但是在设置右对齐时要注意, 其实现原理是通过相对定位和绝对定位来实现下拉菜单的右对齐。首先是对.dropdown 这个 DOM 元素设置了相对定位, 而对应它的.dropdown-menu 子 DOM 元素设置了绝对定位, 而对.dropdown-menu-right 这个 class 类设置了 right:0px 样式属性, 这样下拉菜单就实现了相对于父元素的右对齐效果(仅贴.dropdown 类的 DOM 元素右侧)。

补充: `<span class='caret'></span>` 为 Bootstrap 全局样式的辅助类——三角符号。

13.4.3 Bootstrap的输入框组组件

输入框组是一个复合组件, 基于 Bootstrap 全局表单 CSS 样式(全局 CSS 表单样式将在后面进行详解, 在此并不影响对输入框组的学习), 并且里面可以嵌套 Bootstrap 的按钮组件、下拉菜单组件等, 但是使用时有稍许变化。下面将对输入框组组件进行详细讲解。

实例代码:

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title></title>
8     <link href="css/mybootstrap.css" rel="stylesheet">
9     <style>
10         body{padding:10px 0};
11     </style>
12 </head>
13 <body>
14 <div class="col-xs-3">
15     <div class="input-group">
16         <span class="input-group-addon">@</span>
17         <input type="text" class="form-control" placeholder="Username">
18     </div>
19 </div>
20 <div class="col-xs-3">
21     <div class="input-group">
22         <span class="input-group-addon">
23             <input type="checkbox">
24         </span>
25         <input type="text" class="form-control">
26     </div>
27 </div>
28 <div class="col-xs-3">
29     <div class="input-group">
30         <div class="input-group-btn">
31             <button type="button" class="btn btn-default dropdown-toggle"
                data-toggle="dropdown">Action <span class="caret"></span></button>

```

输入框组基本结构

附加额外元素的多选框和单选框

额外附加下拉菜单的输入框组



```
32      <ul class="dropdown-menu" role="menu">
33          <li><a href="#">Action</a></li>
34          <li><a href="#">Another action</a></li>
35          <li><a href="#">Something else here</a></li>
36          <li class="divider"></li>
37          <li><a href="#">Separated link</a></li>
38      </ul>
39  </div>
40  <input type="text" class="form-control">
41 </div>
42 </div>
43 <div class="col-xs-3">
44     <div class="input-group input-group-lg">
45         <div class="input-group-btn">
46             <button type="button" class="btn btn-default">Left</button>
47             <button type="button" class="btn btn-default">Right</button>
48         </div>
49         <input type="text" class="form-control">
50     </div>
51 </div>
52 <script src="js/jquery-2.2.4.min.js"></script>
53 <script src="js/bootstrap.js"></script>
54 </body>
55 </html>
```

运行上述代码，结果如图 13.6 所示。

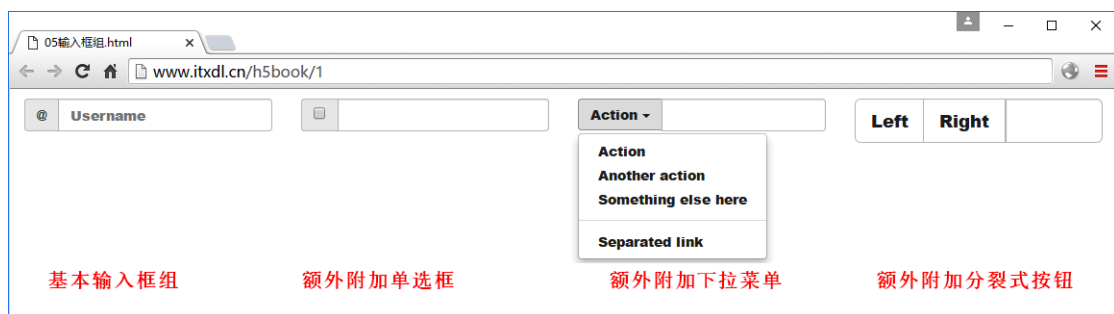


图 13.6 Bootstrap 的输入框组件运行结果

从上述代码可以发现：

- (1) 可以在 `.input-group` 的 DOM 元素中添加对应的 `class` 类来控制输入框组的尺寸。
- (2) 可以在 `.input-group` 下添加 `input-group-addon` 类 DOM 元素，来为输入框组添加额外的文本元素 DOM 标签。
- (3) 可以在 `.input-group` 下添加 `input-group-btn` 类 DOM 元素，来为输入框组添加额外的按钮、按钮组或下拉菜单组件等。

但是注意一点，在使用 `input-group-btn` 类时，按钮组会比表单控件小 1px（在 Chrome 中），如果添加了输入框组尺寸，整个表单组件又能正常显示，但是尺寸只有偏大或偏小，没有中型尺寸怎么办？思路如下，请读者自行尝试完成此处的兼容。

解决方案:

步骤 1: 在 bootstrap/_input-groups.scss 下的第 40 行左右添加一份中型尺寸的 CSS 样式。

步骤 2: 在 bootstrap/_forms.scss 下的第 328 行左右实现一份中型尺寸的混合宏, 然后让步骤 1 的代码来继承此处代码即可 (仿照小/大型尺寸的输入框组代码完成)。

13.4.4 Bootstrap 的导航组件

导航组件也是一个基础组件, 它大致分为两类风格, 一类是基本标签页导航风格, 另一类是胶囊式标签页风格。另外, 其内部也可以嵌套下拉菜单组件, 或者被导航条组件嵌套, 是一类实用性非常强的组件。

实例代码:

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>导航组件</title>
8   <link href="css/mybootstrap.css" rel="stylesheet">
9   <style>
10     body{padding:10px 0};
11   </style>
12 </head>
13 <body>
14 <ul class="nav nav-tabs" role="tablist">
15   <li role="presentation" class="active"><a href="#">Home</a></li>
16   <li role="presentation"><a href="#">Profile</a></li>
17   <li role="presentation" class="dropdown">
18     <a class="dropdown-toggle" data-toggle="dropdown">
19       Dropdown
20       <span class="caret"></span>
21     </a>
22     <ul class="dropdown-menu" role="menu">
23       <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
24         Action</a></li>
25       <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
26         Another action</a></li>
27       <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
28         Something else here</a></li>
29       <li role="presentation"><a role="menuitem" tabindex="-1" href="#">
30         Separated link</a></li>
31     </ul>
32   </li>
33 </ul>
34 <br><br>
35 <ul class="nav nav-pills nav-stacked" role="tablist">
36   <li role="presentation" class="active"><a href="#">Home</a></li>

```

标签页风格的导航

内嵌下拉菜单组件

胶囊式标签页风格导航

导航栏垂直方向堆叠排列



```
33 <li role="presentation" class="disabled"><a href="#">Profile</a></li>
34 <li role="presentation"><a href="#">Messages</a></li>
35 </ul>
36 <script src="js/jquery-2.2.4.min.js"></script>
37 <script src="js/bootstrap.js"></script>
38 </body>
39 </html>
```

禁用的链接

运行实例代码，结果如图 13.7 所示。

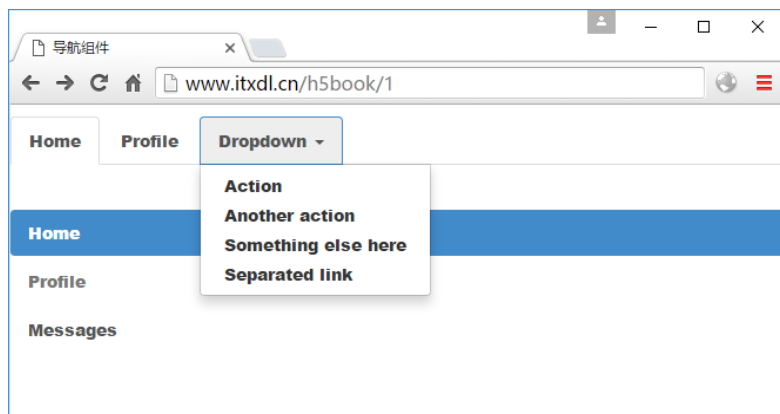


图 13.7 Bootstrap 的导航组件运行结果

运行上述代码可以发现：

- (1) 导航组件最外层 nav 标签，可以通过 nav-tabs 或 nav-pills 类来定义组件是基本标签页导航风格还是胶囊式标签页风格。另外，也可以通过 nav-stacked 类让导航栏堆叠排列。
- (2) 可以通过 dropdown 类，来定义导航栏的某一栏为下拉菜单组件。

13.4.5 顶部工具栏PC端内容填充

学习完上述 Bootstrap 组件，现在笔者将活学活用，来完成实战项目中的顶部工具栏。首先，使用栅格系统把网站 Logo、个人操作导航、搜索和个人状态栏目进行一一划分；其次，分别使用对应组件完成相应的功能模块。

实例代码（PC 端部分代码）：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title></title>
8   <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
```

```

10 <body>
11 <!-- 顶部工具栏 -->
12 <header id="head">
13     <header class="container">
14         <!-- 移动端 -->
15         <div class="hidden-sm hidden-md hidden-lg">移动端</div>
16         <!-- PC端 -->
17         <div class="hidden-xs">
18             <!-- 网站Logo -->
19             <div class="col-sm-2">
20                 
21             </div>
22             <!-- 个人操作导航 -->
23             <div class="col-sm-6 col-md-5 col-lg-4">
24                 <ul class="nav nav-pills" role="tablist">
25                     <li role="presentation" class="active"><a href="#" class="">首页</a></li>
26                     <li role="presentation"><a href="#" class="">我的</a></li>
27                     <li role="presentation"><a href="#" class="">订阅</a></li>
28                     <li role="presentation"><a href="#" class="">会员</a></li>
29                     <li class="dropdown">
30                         <a href="#" data-toggle="dropdown" class="">更多<span class="caret"></span></a>
31                         <ul class="dropdown-menu">
32                             <li><a href="#">更多1</a></li>
33                             <li><a href="#">更多2</a></li>
34                             <li><a href="#">更多3</a></li>
35                         </ul>
36                     </li>
37                 </ul>
38             </div>
39         </div>

```

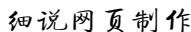
(1) 第 24 行, 使用默认的 nav、nav-pills 组件, 它是 Bootstrap 中的导航组件。

(2) 第 29 行, 使用默认的 dropdown 组件, 它是在 Bootstrap 中的导航组件中再次嵌套一个下拉菜单组件。

```

40 <!-- 搜索 -->
41 <div class="col-sm-1 col-md-2 col-lg-3">
42     <div class="hidden-md hidden-lg">
43         <label class="sr-only" for="topSearch">Email address</label>
44         <input type="email" class="form-control" id="topSearch"
45             placeholder="搜索">
46     </div>
47     <div class="hidden-sm input-group">
48         <input type="text" class="form-control" placeholder="搜索">
49         <span class="input-group-addon"><span class="glyphicon glyphicon-search"></span></span></div>
50 </div>
51

```



(2) 第 46 行，使用默认的 input-group 组件，它是 Bootstrap 中的输入框组组件。

第 54 行和第 55 行，使用默认的 `glyphicon` 组件，其为 Bootstrap 中的字体图标。运行结果如图 13.8~图 13.10 所示。



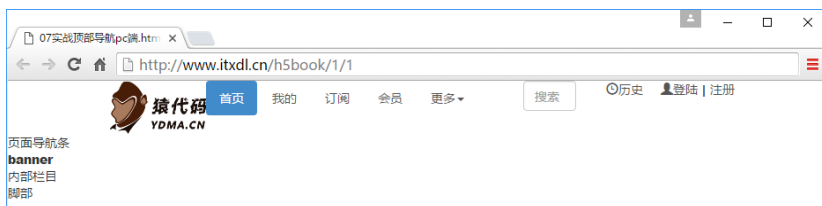


图 13.10 顶部导航 PC 端的小屏显示效果图

运行上述代码，可以发现：

- (1) 网站的 Logo 直接用标签填充。
- (2) 网站的个人操作导航使用了胶囊式标签页导航，并且内部嵌入了下拉菜单（第 29～36 行）。
- (3) 网站的搜索栏通过使用 hidden-sm、hidden-md 和 hidden-lg 三个响应式全局 CSS 样式来进行适配，让小屏幕、中等屏幕和大屏幕进行不同样式的渲染。
- (4) 个人状态信息栏直接使用<a>标签和字体图片来完成。

13.4.6 顶部工具栏PC端样式优化

针对本项目的样式优化，笔者将采用样式覆盖的形式来对页面的样式进行渲染，这样的开发速度会相对较快。但在实际项目开发中，一般 Bootstrap 中的所有全局样式和组件并不会全部使用到，会根据实际需求做相应的删减或修改，并额外添加一些其他的全局样式和组件，这无疑是不适合教学的（工作量太大）。

首先，笔者设置一些常用的全局样式，比如边框、色彩、字体和外边距，其代码如下。

Sass 实战代码（mybootstrap/_variables.scss）：

```

1 /*字体风格*/
2 $general-line-height: 10px !default;
3
4 $line-height-1: $general-line-height * 1 !default;
5 $line-height-2: $general-line-height * 2 !default;
6 $line-height-3: $general-line-height * 3 !default;
7 $line-height-4: $general-line-height * 4 !default;
8 $line-height-5: $general-line-height * 5 !default;
9 $line-height-6: $general-line-height * 6 !default;
10
11 /*内边距和外边距变量的定义*/
12 $general-distance: 5px !default;
13
14 $distance-size-1: $general-distance * 1 !default;
15 $distance-size-2: $general-distance * 2 !default;
16 $distance-size-3: $general-distance * 3 !default;
17 $distance-size-4: $general-distance * 4 !default;
18 $distance-size-5: $general-distance * 5 !default;

```



```
19 $distance-size-6:                $general-distance * 6 !default;
20
21 /*色彩*/
22 $system-gray-dark:                #555 !default;
23 $system-gray-darker:              #777 !default;
24 $system-gray-999:                 #999 !default;
25 $system-gray:                     #ccc !default;
26 $system-gray-lighter:             #ddd !default;
27 $system-gray-light:               #eee !default;
28 $system-white:                    #fff !default;
29 $system_orange:                   orange !default;
30 $system_orange-light:             lighten($system_orange, 10%) !default;
31
32 /*边框*/
33 $general-border-distance:          1px !default;
34
35 $border-distance-size-0:           $general-border-distance * 0 !default;
36 $border-distance-size-1:           $general-border-distance * 1 !default;
37 $border-distance-size-2:           $general-border-distance * 2 !default;
38 $border-distance-size-3:           $general-border-distance * 3 !default;
39 $border-distance-size-4:           $general-border-distance * 4 !default;
40 $border-distance-size-5:           $general-border-distance * 5 !default;
41 $border-distance-size-6:           $general-border-distance * 6 !default;
```

笔者把本项目所需要的变量全部定义到上述的变量文件中（mybootstrap/_variables.scss），后面所有的全局 CSS 样式和组件将从本文件取值。

Sass 实战代码（mybootstrap/mixins/_border.scss）：

```
1 @mixin border-size($direction, $size, $color: $system-gray) {
2   border-style: solid;
3   border-color: $color;
4   border-width: 0;
5   border-#{ $direction }-width: $size;
6 }
7
8 @mixin border-no-size($direction, $size, $color: $system-gray) {
9   border-style: solid;
10  border-color: $color;
11  border-width: $size;
12  border-#{ $direction }-width: 0;
13 }
```

为了完成自定义的全局边框 CSS 样式，笔者定义了两个混合宏：border-size 定义某一个边框的样式；border-no-size 定义三个边框的样式。

Sass 实战代码（mybootstrap/_border.scss）：

```
1 @for $i from 0 through 6 {
2   //四边的边框大小
3   .border-size-#{ $i } {
4     border-width: $border-distance-size-1 * $i;
5   }
6   //单边的边框大小
7   .border-top-size-#{ $i } {
8     @include border-size(top, $border-distance-size-1 * $i);
9   }
}
```

```

10 .border-left-size-#{ $i } {
11     @include border-size(left, $border-distance-size-1 * $i);
12 }
13 .border-right-size-#{ $i } {
14     @include border-size(right, $border-distance-size-1 * $i);
15 }
16 .border-bottom-size-#{ $i } {
17     @include border-size(bottom, $border-distance-size-1 * $i);
18 }
19 //三边边框的大小
20 .border-no-top-size-#{ $i } {
21     @include border-no-size(top, $border-distance-size-1 * $i)
22 }
23 .border-no-left-size-#{ $i } {
24     @include border-no-size(left, $border-distance-size-1 * $i)
25 }
26 .border-no-right-size-#{ $i } {
27     @include border-no-size(right, $border-distance-size-1 * $i)
28 }
29 .border-no-bottom-size-#{ $i } {
30     @include border-no-size(bottom, $border-distance-size-1 * $i)
31 }
32 //边框的颜色
33 .border-gray-color {
34     border-color: $system-gray;
35 }
36 .border-orange-color {
37     border-color: $system-orange;
38 }
39 }

```

通过已经定义好的混合宏和 Sass 变量，现在笔者可直接定义好边框的全局样式，包括一边边框的全局 CSS 样式、三边边框的全局 CSS 样式，还有一些特定色彩边框。

Sass 实战代码（mybootstrap/_color.scss）：

```

1 //灰色
2 .bg-gray-color{
3     background-color:$gray;
4 }
5 //亮灰色
6 .bg-gray-lighter-color{
7     background-color:$gray-lighter;
8 }
9 //系统橘色
10 .bg-orange-color{
11     background-color:$system_orange;
12 }

```

定义一些背景颜色的全局 CSS 样式。如上述所示，笔者不仅使用了自定义的全局变量，而且借用了 Bootstrap 系统库内部默认变量，来设置自定义样式（上述代码的第 3 行和第 7 行）。



Sass 实战代码 (mybootstrap/_font.scss):

```
1 a{
2   color:$gray;
3   text-decoration: none;
4 }
5 a:hover{
6   text-decoration: none;
7 }
8 .fluid-width{
9   width:100%;
10 }
11 /*定义颜色*/
12 .myFont-gray-color{
13   color:$gray;
14 }
15 .myFont-gray-light-color{
16   color:$gray-light;
17 }
18 .myFont-gray-lighter-color{
19   color:$system-gray-999;
20 }
21 /*定义行高*/
22 @for $i from 1 through 6 {
23   .line-height-#{ $i }{
24     line-height: $line-height-1 * $i;
25   }
26 }
```

接着定义一些字体全局 CSS 样式。比如，修改了<a>标签的默认系统样式；定义了一些常用的全局字体颜色样式和字体行高。

Sass 实战代码 (mybootstrap/_paddingOrMargin.scss):

```
1 @for $i from 0 through 6 {
2   //四边的内补白
3   .padding-size-#{ $i } {
4     padding:$distance-size-1 * $i;
5   }
6   //水平的内补白
7   .padding-level-size-#{ $i } {
8     padding:0 $distance-size-1 * $i;
9   }
10  //垂直的内补白
11  .padding-vertical-size-#{ $i } {
12    padding:$distance-size-1 * $i 0;
13  }
14  //单边的内补白
15  .padding-top-size-#{ $i } {
16    padding-top:$distance-size-1 * $i;
17  }
18  .padding-left-size-#{ $i } {
19    padding-left:$distance-size-1 * $i;
20  }
21  .padding-right-size-#{ $i } {
22    padding-right:$distance-size-1 * $i;
23  }
24  .padding-bottom-size-#{ $i } {
25    padding-bottom:$distance-size-1 * $i;
26  }
```

```

27 //四边的外补白
28 .margin-size-#{ $i } {
29     margin: $distance-size-1 * $i;
30 }
31 //水平的外补白
32 .margin-level-size-#{ $i } {
33     margin: 0 $distance-size-1 * $i;
34 }
35 //垂直的外补白
36 .margin-vertical-size-#{ $i } {
37     margin: $distance-size-1 * $i 0;
38 }
39 //单边的外补白
40 .margin-top-size-#{ $i } {
41     margin-top: $distance-size-1 * $i;
42 }
43 .margin-left-size-#{ $i } {
44     margin-left: $distance-size-1 * $i;
45 }
46 .margin-right-size-#{ $i } {
47     margin-right: $distance-size-1 * $i;
48 }
49 .margin-bottom-size-#{ $i } {
50     margin-bottom: $distance-size-1 * $i;
51 }
52 }

```

然后，自定义最常用的一类全局 CSS 样式，即内补白和外补白。从上述代码可以发现，笔者从各个方位定义了相应的内外补白全局 CSS 样式。

Sass 实战代码（mybootstrap/_mystyle.scss）：

```

1 //用于顶部导航的CSS样式
2 .myNav-style-1 {
3     font-size: $font-size-large;
4     font-weight: bold;
5     color: $gray;
6     li {
7         a {
8             color: $gray;
9             padding: $distance-size-2;
10        }
11        &.active a, &.active a:hover{
12            color: $system-white;
13            background-color: $system_orange;
14        }
15        a:hover{
16            color: $system-gray-darker;
17            background-color: $system-gray-light;
18        }
19    }
20 }

```

在此之前，笔者已经定义好了很多通用的全局样式。而_mystyle.scss 文件的作用主要是为项目装饰独有风格的层叠样式，但它与组件的性质不同，其通用性不够强。



而上述的.myNav-style-1 类主要是为了实现个人操作导航的层叠样式覆盖，形成具有独特风格样式的导航。

实例代码（PC 端部分代码）：

```
11 <!-- 顶部工具栏 -->
12 <header id="head" class="border-bottom-size-2 padding-bottom-size-2">
13   <header class="container">
14     <!-- 移动端 -->
15     <div class="hidden-sm hidden-md hidden-lg">移动端</div>
16     <!-- PC端 -->
17     <div class="hidden-xs padding-top-size-1">
18       <!--网站Logo-->
19       <div class="col-sm-2">
20         
21       </div>
```

(1) 第 12 行，使用了自定义的 border-bottom-size-2、padding-bottom-size 全局 CSS 样式，让顶部工具栏的底部有 10px 的内补白和 2px 的底边框。

(2) 第 17 行，使用了自定义的 padding-top-size-1 全局 CSS 样式，使固定容器顶部有 5px 的内补白。

```
22   <!--个人操作导航-->
23   <div class="col-sm-6 col-md-5 col-lg-4">
24     <ul class="nav nav-pills margin-size-2 myNav-style-1" role=
25       "tablist">
26       <li role="presentation" class="active"><a href="#" class=
27         "">首页</a></li>
28       <li role="presentation"><a href="#" class="">我的
29         </a></li>
30       <li role="presentation"><a href="#" class="">订阅
31         </a></li>
32       <li role="presentation"><a href="#" class="">会员
33         </a></li>
34       <li class="dropdown">
35         <a href="#" data-toggle="dropdown" class=
36           "dropdown-toggle">更多<span class="caret"></span></a>
37         <ul class="dropdown-menu">
38           <li><a href="#">更多1</a></li>
39           <li><a href="#">更多2</a></li>
40           <li><a href="#">更多3</a></li>
41         </ul>
42       </li>
43     </ul>
44   </div>
```

第 24 行，使用自定义的 margin-size-2 全局 CSS 样式和 myNav-style-1 组件样式，让个人操作导航显示独有的风格。

```

40      <!--搜索-->
41      <div class="col-sm-1 col-md-2 col-lg-3 margin-vertical-size-3">
42          <div class="hidden-md hidden-lg">
43              <label class="sr-only" for="topSearch">Email address
44              </label>
45              <input type="email" class="form-control" id="topSearch"
46                  placeholder="搜索">
47          </div>
48          <div class="hidden-sm input-group">
49              <input type="text" class="form-control" placeholder=
50                  "搜索">
51              <span class="input-group-addon"><span class="glyphicon
52                  glyphicon-search"></span></span>
53          </div>
54      </div>

```

第 41 行，使用自定义的 `margin-vertical-size-3` 全局 CSS 样式，让搜索栏在固定容器内上下外补白分别为 15px。

[illegible]

(1) 第 53 行，使用了 `margin-vertical-size-3`、`line-height-4` 全局 CSS 样式，让个人状态信息的字体行高为 40px。

(2) 第 54 行, 使用了 myFont--gray-light-color 全局 CSS 样式, 让标签中的字体颜色为亮灰色。第 55 行也同理。

运行结果如图 13.11~图 13.13 所示。



图 13.11 完成 Sass 顶部导航 PC 端的样式优化运行结果——大屏



图 13.12 完成 Sass 顶部导航 PC 端的样式优化运行结果——中等屏



图 13.13 完成 Sass 顶部导航 PC 端的样式优化运行结果——小屏

13.4.7 实战顶部工具栏移动端

下面笔者将完成顶部工具栏移动端的布局，由于移动端的屏幕宽度有限，所以不可能把四项内容在同一行平铺。笔者把个人操作导航放到了屏幕左下侧的下拉菜单中，把搜索和个人状态栏目放到了右侧的下拉菜单中，如下所示。

实例代码（移动端部分代码）：

```
14      <!-- 移动端 -->
15      <div class="hidden-sm hidden-md hidden-lg text-center">
16          <!--头部导航信息-->
17          <div class="col-xs-2 dropdown">
18              <button class="btn btn-default dropdown-toggle h3" type=
19                  "button" id="phone_Navhead"
20                  data-toggle="dropdown">
21                  <span class="glyphicon glyphicon-th-large"></span>
22              </button>
23              <ul class="dropdown-menu text-right" role="menu"
24                  aria-labelledby="phone_Navhead">
25                  <li role="presentation"><a role="menuitem" tabindex="-1"
26                      href="#">首页</a></li>
27                  <li role="presentation"><a role="menuitem" tabindex="-1"
28                      href="#">我的</a></li>
29                  <li role="presentation"><a role="menuitem" tabindex="-1"
30                      href="#">订阅</a></li>
31                  <li role="presentation"><a role="menuitem" tabindex="-1"
32                      href="#">会员</a></li>
33              </ul>
34          </div>
```

```

29      <!--头部标题-->
30      <div class="col-xs-8 h3">猿代码直播</div>
31      <!--头部个人信息-->
32      <div class="col-xs-2 dropdown">
33          <button class="btn btn-default dropdown-toggle h3" type=
34              "button" id="phone_PersonInfo"
35                  data-toggle="dropdown">
36              <span class="glyphicon glyphicon-align-right"></span>
37          </button>
38          <ul class="dropdown-menu dropdown-menu-right" role="menu"
39              aria-labelledby="phone_PersonInfo">
40              <li role="presentation"><a role="menuitem" href="#">搜索
41                  </a></li>
42              <li role="presentation"><a role="menuitem" href="#">登录
43                  </a></li>
44              <li role="presentation"><a role="menuitem" href="#">注册
45                  </a></li>
46              <li role="presentation"><a role="menuitem" href="#">历史
47                  </a></li>
48          </ul>
49      </div>
50  </div>

```

第 17 行和第 32 行，使用了默认的 dropdown 组件，其为 Bootstrap 中的下拉菜单组件。运行结果如图 13.14 所示。



图 13.14 顶部导航的移动端运行结果

可以发现，笔者对移动端的顶部布局没有做过于精细的处理，仅仅只是运用 Bootstrap 的默认组件来完成移动端顶部工具栏的简单布局。当然，读者可以根据个人能力对其进行进一步优化。

13.5 页面导航条

页面导航条是每个网站的必要部分，笔者把其分为两部分：导航条 Logo 部分和内容部



分。当然，Bootstrap 内部也集成了导航条组件，PC 端页面可以直接使用其导航条组件来完成。

但如果布局移动端界面，由于屏幕宽度有限，就需要尽可能留下有用的部分。所以在布局移动端界面时，笔者把导航条的 Logo 部分进行隐藏，把导航条的内容部分直接使用栅格系统平铺至整行。

首先还是把移动端和 PC 端的内容使用响应式进行隔离，如下所示：

```
89 <div class="clearfix"></div>
90 <!-- 页面导航条 -->
91 <nav id="pageNav">
92     <nav class="container">
93         <div class="hidden-xs">pc端</div>
94         <div class="hidden-sm hidden-md hidden-lg text-center">移动端</div>
95     </nav>
96 </nav>
```

其中第 89 行为 Bootstrap 的全局 CSS 样式，其主要功能为清除浮动。

而#pageNav 和#pageNav Nav 这两个 DOM 节点的关系与顶部工具栏同理。前者负责整个界面导航条的背景颜色、边框等，而后者则主要是内容填充。

13.5.1 Bootstrap 的导航条组件

Bootstrap 的导航条是一个高级组件，它里面可以嵌套很多其他组件，比如下拉菜单、按钮组、输入框组、表单控件和导航等。而其自身也有很多样式特性，比如可以设置导航条头部；可以对导航条组件进行排列；可以设置其 fixed 定位在浏览器的顶部和底部；也可以设置反色导航条。另外，还可以通过 JavaScript 来折叠导航条，使得.navbar-collapse 的 DOM 元素下的节点暂时隐藏。

实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>导航条组件</title>
8     <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11     <nav class="navbar navbar-inverse navbar-fixed-bottom" role="navigation">
12         <div class="container-fluid">
13             <!-- 设置导航条的头部 -->
14             <div class="navbar-header">
15                 <button type="button" class="navbar-toggle collapsed" data-toggle=
16                     "collapse" data-target="#bs-example-navbar-collapse-1">
17                     <span class="sr-only">Toggle navigation</span>
```

```

17     <span class="icon-bar"></span>
18     <span class="icon-bar"></span>
19     <span class="icon-bar"></span>
20 </button>
21 <a class="navbar-brand" href="#">Brand</a>
22 </div>
23 <!-- 被JavaScript控制的折叠隐藏导航元素 -->
24 <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
25     <!-- 普通导航元素 -->
26     <ul class="nav navbar-nav">
27         <li class="active"><a href="#">Link</a></li>
28         <p class="navbar-text">text<a href="#" class="navbar-link">anchor
29         </a></p>
30         <li><a href="#">Link</a></li>
31         <li class="dropdown">
32             <a href="#" class="dropdown-toggle" data-toggle="dropdown">
33                 Dropdown <span class="caret"></span></a>
34             <ul class="dropdown-menu" role="menu">
35                 <li><a href="#">Action</a></li>
36                 <li><a href="#">Another action</a></li>
37                 <li><a href="#">Something else here</a></li>
38                 <li class="divider"></li>
39                 <li><a href="#">Separated link</a></li>
40                 <li class="divider"></li>
41                 <li><a href="#">One more separated link</a></li>
42             </ul>
43         </li>
44     </ul>
45     <!-- 导航的表单控件元素 -->
46     <form class="navbar-form navbar-left" role="search">
47         <div class="form-group">
48             <input type="text" class="form-control" placeholder="Search">
49         </div>
50         <button type="submit" class="btn btn-default">Submit</button>
51     </form>
52     <!-- 普通导航元素，并向右对齐 -->
53     <ul class="nav navbar-nav navbar-right">
54         <li><a href="#">Link</a></li>
55         <li class="dropdown">
56             <a href="#" class="dropdown-toggle" data-toggle="dropdown">
57                 Dropdown <span class="caret"></span></a>
58             <ul class="dropdown-menu" role="menu">
59                 <li><a href="#">Action</a></li>
60                 <li><a href="#">Another action</a></li>
61                 <li><a href="#">Something else here</a></li>
62                 <li class="divider"></li>
63                 <li><a href="#">Separated link</a></li>
64             </ul>
65         </li>
66     </ul>
67 </div><!-- /.navbar-collapse -->
68 </div><!-- /.container-fluid -->
69 </nav>

```

被JavaScript控制折叠的DOM元素

非导航的链接

表单导航控件

左对齐

右对齐

运行结果如图 13.15 所示。

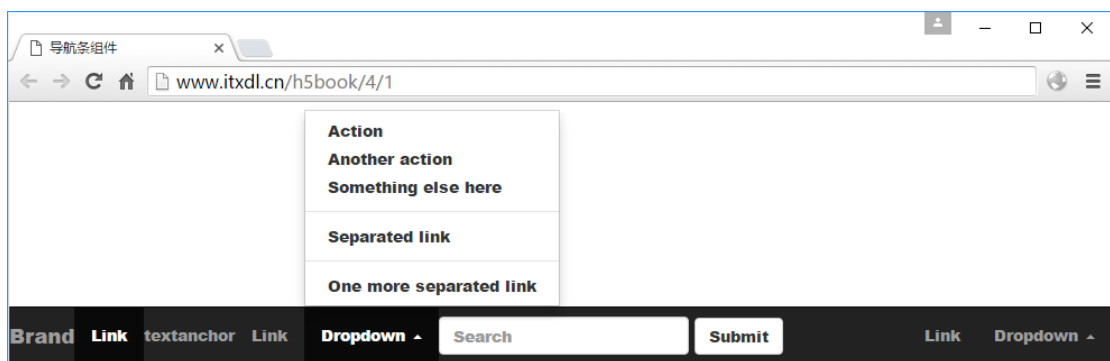


图 13.15 Bootstrap 的导航条组件运行结果

上述代码讲述了 Bootstrap 的导航条组件中常用的功能，当然读者也可以灵活多变，写出独有风格的导航条，比如下面实战页面中的导航条。

13.5.2 实战页面导航条——内容填充

页面导航条分为两部分，即导航条 Logo 部分和导航条内容部分。在 PC 端布局时，直接使用 Bootstrap 的导航条组件进行全部显示；在移动端布局时，使用栅格系统把导航条的内容部分平铺到一列上。

实战代码（页面导航条内容填充）：

```
89 <div class="clearfix"></div>
90 <!-- 页面导航条 -->
91 <nav id="pageNav">
92     <nav class="container">
93         <div class="navbar navbar-default hidden-xs" role="navigation">
94             <div class="navbar-header">
95                 <a href="#" class="navbar-brand"><img src=
96                     "images/time.png" class="logo" alt="logo" height="50px"></a>
97             </div>
98             <ul class="nav navbar-nav">
99                 <li><a href="#">直播首页</a></li>
100                <li><a href="#">正在直播</a></li>
101                <li class="active"><a href="#">即将直播</a></li>
102                <li><a href="#">精彩回放</a></li>
103            </ul>
104            <div class="hidden-sm hidden-md hidden-lg text-center">
105                <div class="col-xs-3"><a href="#">直播首页</a></div>
106                <div class="col-xs-3 active"><a href="#">正在直播</a></div>
107                <div class="col-xs-3"><a href="#">即将直播</a></div>
108                <div class="col-xs-3"><a href="#">精彩回放</a></div>
109            </div>
110        </nav>
111 </nav>
```

第 93 行，使用了默认的 navbar 组件，其为 Bootstrap 中的导航条组件。

运行结果如图 13.16~图 13.18 所示。

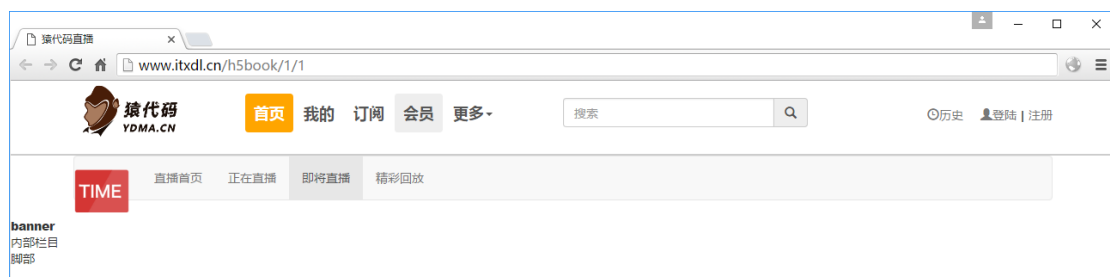


图 13.16 实战页面导航条——内容填充大屏幕运行结果



图 13.17 实战页面导航条——内容填充小屏幕运行结果



图 13.18 实战页面导航条——内容填充超小屏幕运行结果

由上可以发现，经过内容填充后基本成型，但是样式非常不协调。接下来，对其样式进行优化处理。



13.5.3 实战页面导航条——样式优化

同顶部工具栏的样式优化一样，已经设置好了常用的全局样式，但网站某些组件的独有风格样式还需要由开发人员自己定义，如下所示。

Sass 实战代码（mybootstrap/_mystyle.scss）：

```
21 /*页面导航条的CSS样式*/
22 .myNav-style-2{
23   //PC端
24   ul.navbar-nav li{
25     a{
26       color: $gray;
27       border-bottom: $border-distance-size-5 solid $system-gray-light;
28     }
29     &.active a, &.active a:hover{
30       color: $system_orange;
31       background-color: $system_gray-light;
32       border-color: $system_orange;
33     }
34     a:hover{
35       font-weight: bold;
36       color: $system_orange-light;
37     }
38   }
39   //移动端
40   div.hidden-sm div.active a{
41     color: $system_orange;
42   }
43 }
```

配合 Bootstrap 的导航条组件，使用 Sass 对其样式进行覆盖，形成独有风格的导航条样式组件。

实战代码：

```
89 <div class="clearfix"></div>
90 <!--页面导航条-->
91 <nav id="pageNav" class="bg-gray-lighter-color">
92   <nav class="container padding-vertical-size-2 myNav-style-2">
93     <div class="navbar navbar-default hidden-xs margin-bottom-size-0
94       border-size-0 bg-gray-lighter-color " role="navigation">
95       <div class="navbar-header padding-left-size-4">
96         <a href="###" class="navbar-brand padding-size-0"><img src=
97           "images/time.png" class="logo" alt="logo" height="50px"></a>
98       </div>
99       <ul class="nav navbar-nav">
100         <li><a href="###">直播首页</a></li>
101         <li><a href="###">正在直播</a></li>
102         <li class="active"><a href="###">即将直播</a></li>
103         <li><a href="###">精彩回放</a></li>
104       </ul>
105     </div>
106     <div class="hidden-sm hidden-md hidden-lg text-center">
```

```

105     <div class="col-xs-3"><a href="">直播首页</a></div>
106     <div class="col-xs-3 active"><a href="">正在直播</a></div>
107     <div class="col-xs-3"><a href="">即将直播</a></div>
108     <div class="col-xs-3"><a href="">精彩回放</a></div>
109   </div>
110 </nav>
111 </nav>

```

(1) 第 91 行, 使用了自定义的 `bg-gray-lighter-color` 全局 CSS 样式, 让其整个导航条背景颜色为灰白色。

(2) 第 92 行, 使用了自定义的 `padding-vertical-size-2` 全局 CSS 样式和 `myNav-style-2` 独有风格组件样式。在 PC 端中直接使用 `navbar` 加上 `myNav-style-2` 共同渲染页面导航条层叠样式。

运行结果如图 13.19~图 13.22 所示。



图 13.19 页面导航条样式优化的运行结果——大屏



图 13.20 页面导航条样式优化的运行结果——中等屏



图 13.21 页面导航条样式优化的运行结果——小屏



图 13.22 页面导航条样式优化的运行结果——超小屏

如上所示，完成了页面导航条功能。其中可以发现，笔者通过 `mybootstrap/_mystyle.scss` 文件自定义了独特风格的页面导航条，也请读者认真思考 `.myNav-style-1` 和 `.myNav-style-2` 这两个类的运用思路，如果在真正的项目开发中，又该如何去设置组件的样式风格？

13.6 banner区

实战项目 banner 区（横幅）功能模块，是页面中最醒目的一个功能模块，在这里笔者直接使用 Bootstrap 默认的轮播图。不同于上面两个功能模块，横幅模块不进行 PC 端和移动端的区块划分，直接用 `width: 100%` 进行平铺整行。

13.6.1 Bootstrap的JavaScript轮播图插件

Bootstrap 默认的 JavaScript 轮播图插件非常简单好用，它通过内部 JavaScript 作用，基于配置来设置其轮播图的各个功能参数，比如轮播图的播放顺序、播放的事件间隔、是否循环播放轮播图等。

实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>猿代码直播</title>
8   <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11   <div id="carousel" class="carousel slide" data-ride="carousel" data-pause=
    "false" data-wrap="true" data-interval="2000">
12     <!-- 轮播图平滑滑动 当鼠标悬浮时，是否停止播放 -->
13     <ol class="carousel-indicators">
14       <li data-target="#carousel" data-slide-to="0" class="active"></li>
```

```

16     <li data-target="#carousel" data-slide-to="2"></li>
17 </ol>
18 <!-- 轮播图片播放区 -->
19 <div class="carousel-inner" role="listbox">
20     <div class="item active">
21         <a href=""></a>
22         <div class="carousel-caption">
23             <h4>标题一</h4>
24             <p>图片一内容简介</p>
25         </div>
26     </div>
27     <div class="item">
28         <a href=""></a>
29         <div class="carousel-caption">
30             <h4>标题二</h4>
31             <p>图片二内容简介</p>
32         </div>
33     </div>
34     <div class="item">
35         <a href=""></a>
36         <div class="carousel-caption">
37             <h4>标题三</h4>
38             <p>图片三内容简介</p>
39         </div>
40     </div>
41 </div>
42 <!-- 轮播图片控制器 -->
43 <a class="left carousel-control" href="#carousel" role="button"
44     data-slide="prev">
45     <span class="glyphicon glyphicon-chevron-left"></span>
46     <span class="sr-only">Previous</span>
47 </a>
48 <a class="right carousel-control" href="#carousel" role="button"
49     data-slide="next">
50     <span class="glyphicon glyphicon-chevron-right"></span>
51     <span class="sr-only">Next</span>
52 </a>
53 </div>
54 <script src="js/jquery-2.2.4.min.js"></script>
55 <script src="js/bootstrap.js"></script>
56 </body>
57 </html>

```

运行结果如图 13.23 所示。

Bootstrap 的轮播图插件代码比较固定，如上所示，整个轮播图分为三部分——图片计数器、图片播放区和图片控制器。而此三部分又被 `carousel` 类的 DOM 对象容器包裹着。

其中轮播图的容器 DOM 对象可以设置各种特性参数，比如 `class="slide"` 用来设置轮播图图片之间的平滑滑动；`data-interva="5000"` 用来设置轮播图播放的时间间隔（默认为 5 秒）；`data-wrap="true"` 用来设置轮播图是否循环播放（默认为 `true`）；`data-pause="false"` 表示当鼠标停留在轮播图上时，是否停止轮播（默认值为 `hover`，表示停止播放）。

而其他的图片计数器、图片播放区和图片控制器，请读者参考官方手册进行详细学习。



图 13.23 Bootstrap 的 JavaScript 轮播图插件运行结果

13.6.2 实战banner区

在此，笔者直接使用 Bootstrap 的轮播图插件的默认样式完成实战项目的 banner 模块，如下所示。

实战代码：

```
112 <!-- banner -->
113 <div id="carousel" class="carousel slide" data-ride="carousel">
114   <!-- 轮播图片计数器 -->
115   <ol class="carousel-indicators">
116     <li data-target="#carousel" data-slide-to="0" class="active"></li>
117     <li data-target="#carousel" data-slide-to="1"></li>
118     <li data-target="#carousel" data-slide-to="2"></li>
119   </ol>
120   <!-- 轮播图片播放区 -->
121   <div class="carousel-inner" role="listbox">
122     <div class="item active">
123       <a href=""></a>
124       <div class="carousel-caption">
125         <h4>标题一</h4>
126         <p>图片一内容简介</p>
127       </div>
128     </div>
129     <div class="item">
130       <a href=""></a>
131       <div class="carousel-caption">
132         <h4>标题二</h4>
133         <p>图片二内容简介</p>
134       </div>
135     </div>
136     <div class="item">
137       <a href=""></a>
138       <div class="carousel-caption">
139         <h4>标题三</h4>
```

```

140         <p>图片三内容简介</p>
141     </div>
142 </div>
143 </div>
144 <!-- 轮播图片控制器 -->
145 <a class="left carousel-control" href="#carousel" role="button"
146     data-slide="prev">
147     <span class="glyphicon glyphicon-chevron-left"></span>
148 </a>
149 <a class="right carousel-control" href="#carousel" role="button"
150     data-slide="next">
151     <span class="glyphicon glyphicon-chevron-right"></span>
152 </a>
153 </div>

```

第 113 行，使用了默认的.carousel JavaScript 插件，其为 Bootstrap 中的 JavaScript 轮播图插件。

运行结果如图 13.24 和图 13.25 所示。



图 13.24 实战 banner 区 PC 端运行结果



图 13.25 实战 banner 区移动端运行结果



13.7 推荐位

在项目中，推荐位的风格多种多样，而每种风格的推荐位在 PC 端和移动端又有不同风格的布局。在内容层面上，推荐位的内容又划分为头部导航部分和内容部分。

而在本实战项目中，笔者写了两种风格的推荐位，如下所示：

```
154 <!-- 推荐位 -->
155 <div id="pageContainer">
156     <section class="container">
157         第一种风格的推荐位
158     </section>
159     <section class="container">
160         第二种风格的推荐位
161     </section>
162 </div>
```

其中，第一种风格的推荐位功能模块基本布局思路为：推荐位的头部使用 Bootstrap 的导航完成，并且里面还包括推荐位导航 Logo；推荐位的内容部分使用 Bootstrap 的缩略图组件来完成。

而第二种风格的推荐位功能模块布局思路为：推荐位的头部使用 Bootstrap 的导航完成，同上；而推荐位的内容部分则使用 Bootstrap 的面板组件来完成。

13.7.1 Bootstrap的缩略图组件

Bootstrap 的缩略图组件是一个基础组件，使用非常简单，其基本用法如下。

实例代码：

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>bootstrap的缩略图</title>
8     <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11 <div class="col-xs-6 col-sm-4">
12     <div class="thumbnail">
13         
15         <div class="caption">
16             <h5>架构师之路 </h5>
17             <p>1000人正在观看</p>
18         </div>
```



```

19     </div>
20 </div>
21 <script src="js/jquery-2.2.4.min.js"></script>
22 <script src="js/bootstrap.js"></script>
23 </body>
24 </html>

```

运行结果如图 13.26 所示。

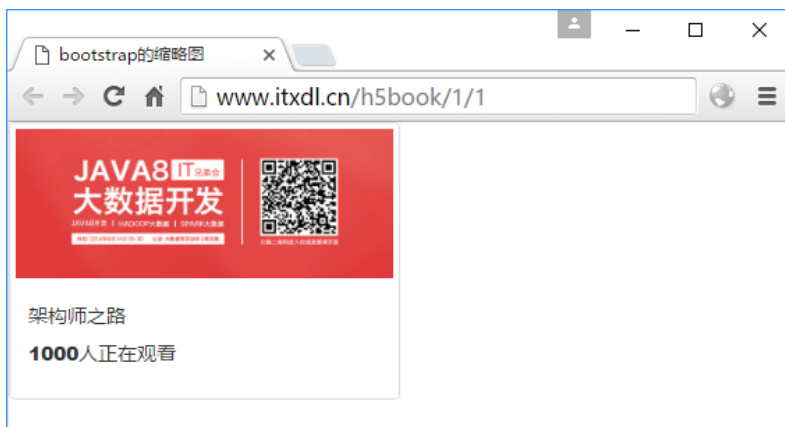


图 13.26 Bootstrap 的缩略图组件运行结果

13.7.2 Bootstrap的面板组件

面板在项目中也是常用的一种功能组件，而 Bootstrap 的面板组件内部功能也比较丰富，包括头部、内容部分和尾部，也可以设置其情景效果，并且内容部分可以内嵌表格或者列表。实例如下所示。

实例代码：

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>猿代码直播</title>
8     <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11 <div class="panel panel-success">
12     <div class="panel-heading">面板的头部</div>
13     <table class="table">
14         <tr>
15             <td>06:00</td>
16             <td>架构师/产品经理</td>
17             <td>44人订阅</td>

```

面板的情景效果

面板的头部

嵌套表格



```
18         <td><button type="button" class="btn btn-default">立即预约
19         </button></td>
20     </tr>
21     <tr>
22         <td>06:00</td>
23         <td>架构师/产品经理</td>
24         <td>44人订阅</td>
25         <td><button type="button" class="btn btn-default">立即预约
26         </button></td>
27     </tr>
28 </table>
29 <div class="panel-body">面板的内容部分
30 </div>
31 <div class="panel-footer">面板的尾部</div>
32 </div>
33 <script src="js/jquery-2.2.4.min.js"></script>
34 <script src="js/bootstrap.js"></script>
35 </body>
36 </html>
```

面板的内容

面板的尾部

运行结果如图 13.27 所示。

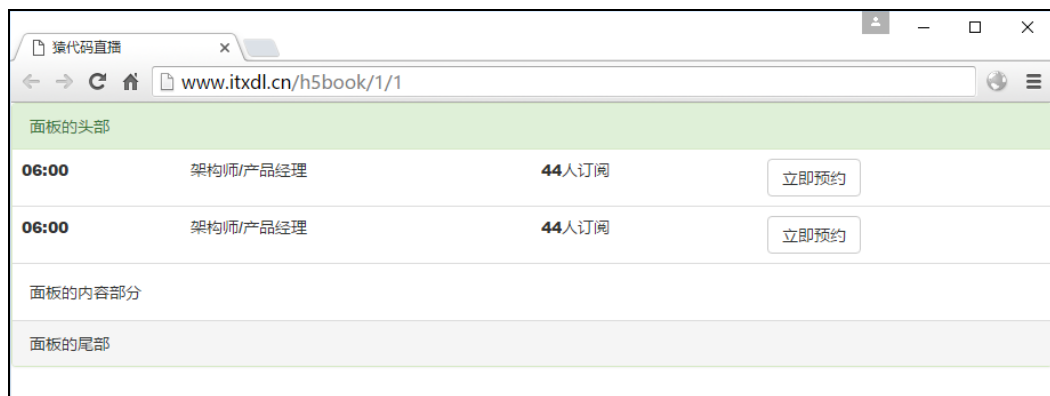


图 13.27 Bootstrap 的面板组件运行结果

可以看到，情景效果对面板的脚部没有样式渲染。在此只演示了面板的基础内容和嵌套表格，嵌套列表请大家自行学习。

13.7.3 实战第一种风格的推荐位——内容填充

第一种风格的推荐位功能模块大致分为两部分——头部和内容部分，其中头部包括推荐位的 Logo 与标题和推荐位的导航，由于移动端的屏幕宽度大小有限，可以把头部的推荐位导航（不包括 Logo 与标题）进行隐藏；而内容部分则直接使用栅格系统进行划分，但要注意移动端和 PC 端的内外边距。

实战代码:

```

154 <!-- 推荐位 -->
155 <div id="pageContainer">
156   <section class="container">
157     <!--PC端内容导航-->
158     <div class="hidden-xs">
159       <ul class="nav nav-tabs " role="tablist">
160         <li role="presentation">
161           <h3 class="">正在直播</h3>
163         </li>
164         <li role="presentation" class="active"><a href="#">最新
165         </a></li>
166         <li role="presentation"><a href="#">最热</a></li>
167       </ul>
168     </div>

```

第 159 行, 使用了默认的 nav、nav-tabs 组件, 其为 Bootstrap 中的导航组件。

```

167 <!-- 移动端内容导航-->
168 <div class="hidden-sm hidden-md hidden-lg">
169   <ul class="nav nav-tabs " role="tablist">
170     <li role="presentation">
171       <h3 class="">正在直播</h3></li>
173     </ul>
174   </div>
175   <div class="col-xs-6 col-sm-4">
176     <div class="thumbnail col-xs-12 col-sm-11">
177       
179       <div class="caption">
180         <h5>架构师之路 </h5>
181         <p class="hidden-xs">1000人正在观看</p>
182         <footer class="hidden-xs">2016-09-13 09:45:03</footer>
183       </div>
184     </div>

```

第 175 行, 使用了默认的 thumbnail 组件, 其为 Bootstrap 中的缩略图组件。

```

185   <div class="col-xs-6 col-sm-4">
186     <div class="thumbnail col-xs-12 col-sm-11">
187       
189       <div class="caption">
190         <h5>架构师之路 </h5>
191         <p class="hidden-xs">1000人正在观看</p>
192         <footer class="hidden-xs">2016-09-13 09:45:03</footer>
193       </div>
194     </div>
195   </div>
196   <div class="col-xs-6 col-sm-4">
197     <div class="thumbnail col-xs-12 col-sm-11">
198       
200       <div class="caption">

```



```
201 <h5>架构师之路 </h5>
202 <p class="hidden-xs">1000人正在观看</p>
203 <footer class="hidden-xs">2016-09-13 09:45:03</footer>
204 </div>
205 </div>
206 </div>
207 </section>
208 <section class="container">
209 </section>
210 </div>
```

运行结果如图 13.28 和图 13.29 所示。

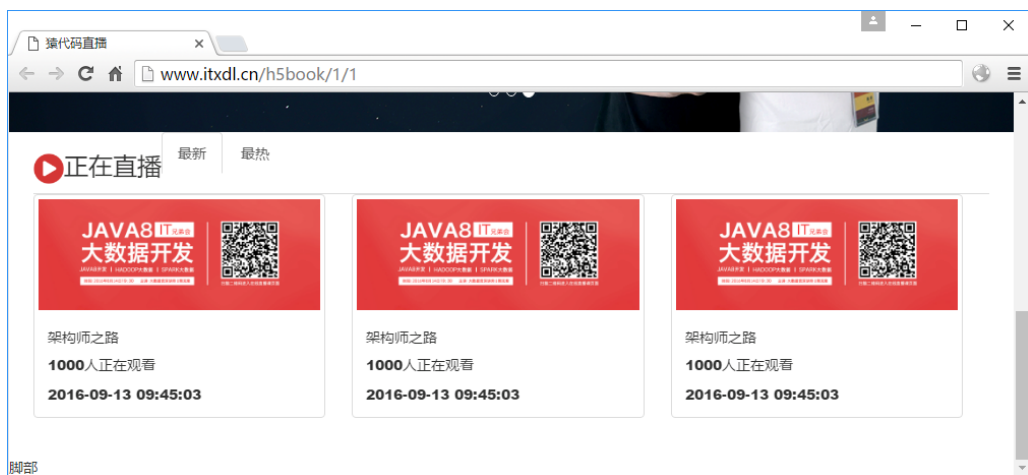


图 13.28 第一种风格的推荐位的内容填充——PC 端运行结果



图 13.29 第一种风格的推荐位的内容填充——移动端运行结果

运行时可以发现，在 PC 端，推荐位的头部导航的 Logo 与标题把导航撑开了，需要进一步调节其样式；而其内容部分，在 PC 端是三个缩略图为一排，并且在.thumbnail 的 DOM 对象中运用了.col-xs-12 与.col-sm-11 进行栅格排列，形成外补白效果。而在移动端则是两个缩略图为一排，并且紧紧相连（无外补白效果）。

注意：在推荐位的内容中，使用.col-xs-12 与.col-sm-11 达到外补白效果是有一定缺陷的——内容不是居中对称显示。由于教学，不能让细节部分过于烦琐，请读者自行尝试达到居中对称效果。

13.7.4 实战第一种风格的推荐位——样式优化

在上一小节中，可以发现推荐位的头部样式已经开始凌乱。现在对此进行单独的样式优化，具体内容如下所示。

Sass 实战代码（mybootstrap/_mystyle.scss）：

```
50 /*推荐位--第一种风格*/
51 .myContains-style-1 .containerNav{
52   border-color:$system-gray;
53   h3{
54     height:40px;
55   }
56   li.active a{
57     background-color:$system-gray-light;
58     border-color:$system-gray;
59     border-bottom-color: $system-gray-light;
60     color:$system_orange;
61   }
62   li a:hover{
63     background-color:$system-gray-lighter;
64   }
65 }
```

上述代码主要是为了调整推荐位头部导航条样式。接下来，看看其具体样式优化代码。

实战代码：

```
154 <!-- 推荐位 -->
155 <div id="pageContainer" class="bg-gray-lighter-color">
156   <section class="container myContains-style-1">
157     <!--PC端内容导航-->
158     <div class="hidden-xs padding-vertical-size-4">
159       <ul class="nav nav-tabs containerNav" role="tablist">
160         <li role="presentation">
161           <h3 class="margin-level-size-4"><img src=
162             "images/logo01.png" width="30" height="30">正在直播</h3>
163         </li>
164         <li role="presentation" class="active"><a href="#">最新
165         </a></li>
166         <li role="presentation"><a href="#">最热</a></li>
167       </ul>
168     </div>
169   </section>
170 </div>
```



(1) 第 155 行, 使用了自定义的 `bg-gray-lighter-color` 全局 CSS 样式, 让推荐位的背景颜色为亮灰色。

(2) 第 156 行, 使用了自定义的 `myContains-style-1` 组件样式。其主要是为了渲染不同风格样式的推荐位, 而不污染其他样式。

(3) 第 158 行, 使用了自定义的 `padding-vertical-size-4` 全局 CSS 样式。

(4) 第 159 行, 使用了自定义的 `containerNav` 组件样式。其主要是为了渲染不同风格的推荐位导航样式。

(5) 第 161 行, 使用了自定义的 `margin-level-size-4` 全局 CSS 样式。

```
167      <!-- 移动端内容导航-->
168      <div class="hidden-sm hidden-md hidden-lg padding-vertical-size-4">
169          <ul class="nav nav-tabs containerNav" role="tablist">
170              <li role="presentation">
171                  <h3 class="margin-level-size-4"><img src=
                    "images/logo01.png" width="30" height="30">正在直播
                  </h3></li>
172      </ul>
173      </div>
```

(1) 第 168 行, 使用了自定义的 `padding-vertical-size-4` 全局 CSS 样式。

(2) 第 169 行, 使用了自定义的 `containerNav` 组件样式。

(3) 第 171 行, 使用了自定义的 `margin-level-size-4` 全局 CSS 样式。

```
174      <div class="col-xs-6 col-sm-4">
175          <div class="thumbnail col-xs-12 col-sm-11">
176              
178              <div class="caption">
179                  <h5>java8大数据 </h5>
180                  <p class="hidden-xs">1000人正在观看</p>
181                  <footer class="hidden-xs border-top-size-1
                    myFont-gray-lighter-color line-height-3">2016-09-13
                    09:45:03</footer>
182              </div>
183          </div>
184      </div>
```

第 181 行, 使用了自定义的 `border-top-size-1`、`myFont-gray-lighter-color`、`line-height-3` 全局 CSS 样式。下面第 192 行和第 203 行同理。

```
185      <div class="col-xs-6 col-sm-4">
186          <div class="thumbnail col-xs-12 col-sm-11">
187              
189              <div class="caption">
190                  <h5>java8大数据 </h5>
191                  <p class="hidden-xs">1000人正在观看</p>
192                  <footer class="hidden-xs border-top-size-1
                    myFont-gray-lighter-color line-height-3">2016-09-13
                    09:45:03</footer>
193              </div>
```

```

194         </div>
195     </div>
196     <div class="col-xs-6 col-sm-4">
197         <div class="thumbnail col-xs-12 col-sm-11">
198             
200             <div class="caption">
201                 <h5>java8大数据 </h5>
202                 <p class="hidden-xs">1000人正在观看</p>
203                 <footer class="hidden-xs border-top-size-1
204                     myFont-gray-lighter-color line-height-3">2016-09-13
205                     09:45:03</footer>
206             </div>
207         </div>
208     </div>
209 </section>
210 <section class="container">
211 </section>
212 </div>

```

运行结果如图 13.30 和图 13.31 所示。

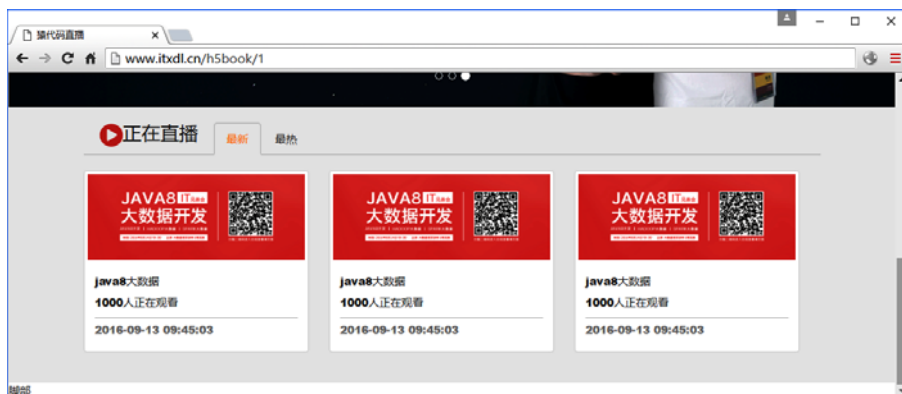


图 13.30 第一种风格的推荐位的样式优化——PC 端运行结果



图 13.31 第一种风格的推荐位的样式优化——移动端运行结果



13.7.5 实战第二种风格的推荐位——内容填充

在第二种风格的推荐位功能模块中，首先按照内容划分，左侧为时间表，右侧为个人订阅信息。当在移动端访问时，把个人订阅信息进行隐藏，把时间表导航在推荐位头部导航中显示；当在 PC 端访问时，则把时间表导航放在面板的头部进行显示。具体代码如下所示。

实例代码：

```
208      <!-- 第二种风格的推荐位 -->
209      <section class="container">
210          <nav class="navbar navbar-default" role="navigation">
211              <div class="navbar-header">
212                  <button type="button" class="navbar-toggle collapsed"
213                      data-toggle="collapse" data-target=
214                          "#bs-navbar-collapse-twoStyle">
215                      <span class="sr-only">Toggle navigation</span>
216                      <span class="icon-bar"></span>
217                      <span class="icon-bar"></span>
218                      <span class="icon-bar"></span>
219                  </button>
220                  <div><h3 class="">直播课时间表</h3></div>
222              </div>
```

第 210 行，使用了默认的 navbar 组件，其为 Bootstrap 的默认导航条组件。

```
220      <div class="collapse navbar-collapse" id=
221          "bs-navbar-collapse-twoStyle">
222          <div class="hidden-sm hidden-md hidden-lg">
223              <ul class="nav navbar-nav">
224                  <li><a href="#">今天 -- 09月21日</a></li>
225                  <li><a href="#">星期二 -- 09月21日</a></li>
226                  <li class="active"><a href="#">星期三 -- 09月22日
227                      </a></li>
228                  <li><a href="#">星期四 -- 09月23日</a></li>
229                  <li><a href="#">星期五 -- 09月24日</a></li>
230                  <li><a href="#">星期六 -- 09月25日</a></li>
231                  <li><a href="#">星期日 -- 09月26日</a></li>
232              </ul>
233          </div>
234      </div>
235      <div class="panel panel-default col-sm-12 col-md-9 col-lg-7">
236          <div class="panel-heading hidden-xs pull-left fluid-width">
237              <ul class="nav navbar-nav">
238                  <li><a href="#">今天 <br> 09月21日</a></li>
239                  <li><a href="#">星期二 <br> 09月21日</a></li>
240                  <li class="active"><a href="#">星期三 <br> 09月22日
241                      </a></li>
242                  <li><a href="#">星期四 <br> 09月23日</a></li>
243                  <li><a href="#">星期五 <br> 09月24日</a></li>
244                  <li><a href="#">星期六 <br> 09月25日</a></li>
```

```

243         </ul>
244     </div>
245     <div class="clearfix"></div>
246     <table class="table">
247         <tr>
248             <td>06:00</td>
249             <td>架构师/产品经理</td>
250             <td class="text-right">143244人订阅</td>
251             <td><button type="button" class="btn btn-default">
                立即预约</button></td>
252         </tr>
253         <tr>
254             <td>06:00</td>
255             <td>架构师/产品经理</td>
256             <td class="text-right">44人订阅</td>
257             <td><button type="button" class="btn btn-default">
                立即预约</button></td>
258         </tr>
259         <tr>
260             <td>06:00</td>
261             <td>架构师/产品经理</td>
262             <td class="text-right">4人订阅</td>
263             <td><button type="button" class="btn btn-default">
                立即预约</button></td>
264         </tr>
265     </table>
266 </div>

```

第 234 行，使用了默认的 panel 组件，其为 Bootstrap 的面板组件。

```

267     <!-- 右侧面板 -->
268     <div class="panel panel-default hidden-xs hidden-sm col-md-3
269         col-lg-push-1 col-lg-4">
270         <div class="panel-heading">
271             <h3>我的订阅</h3>
272         </div>
273         <table class="table">
274             <tr>
275                 <td>09/15 06:00</td>
276                 <td>架构师/产品经理</td>
277             </tr>
278             <tr>
279                 <td>09/15 06:00</td>
280                 <td>架构师/产品经理</td>
281             </tr>
282             <tr>
283                 <td>09/15 06:00</td>
284                 <td>架构师/产品经理</td>
285             </tr>
286         </table>
287     </div>
288 </section>
289 </div>

```

运行结果如图 13.32 和图 13.33 所示。



图 13.32 实战第二种风格的推荐位的内容填充——大屏运行结果



图 13.33 实战第二种风格的推荐位的内容填充——超小屏运行结果

运行上述代码可以发现，网页内容部分已经全部显现出来了，但其样式还有待优化。

注意：PC 端的时间导航存在于面板的头部，而移动端的时间导航存在于推荐位导航的折叠块中。

13.7.6 实战第二种风格的推荐位——样式优化

在上一小节中，笔者使用面板完成第二种风格的推荐位，并且内容也相对较多。而笔者也采用了 Sass 对其进行样式优化，代码如下所示。

Sass 实战代码 (mybootstrap/_mystyle.scss):

```

66 /*推荐位--第二种风格*/
67 .myContains-style-2{
68   //导航
69   nav{
70     border-radius: 0;
71     .navbar-collapse li{
72       a{
73         text-align: center;
74       }
75       &.active a{
76         background-color:$system_orange;
77       }
78     }
79   }
80   //面板内容
81   div.panel{
82     .panel-heading li{
83       a{
84         border-bottom: 3px solid #f5f5f5;
85       }
86       &.active a{
87         color:orange;
88         border-bottom: 3px solid $system_orange;
89       }
90     }
91     table{
92       margin: 0 2%;
93       width: 96%;
94       tr td{
95         border: 0px;
96         border-bottom: 1px solid #ccc;
97         line-height: 50px;
98       }
99       tr:last-child td{
100         border: 0px;
101       }
102       button.btn{
103         background: $system_orange;
104       }
105     }
106   }
107   //左侧面板
108   div.right{
109     .panel-heading h3{
110       line-height: 43px;
111     }
112     table tr td{
113       white-space:nowrap;
114       &:nth-child(2){

```



```
115     width:60%;  
116   }  
117 }  
118 }  
119 }
```

由于第二种风格的推荐位内容比较多,在设置独特的样式时,需要大量的样式进行重叠覆盖原有的样式。

实战代码:

```
208 <!-- 第二种风格的推荐位 -->  
209 <section class="container myContains-style-2">  
210   <nav class="navbar navbar-default bg-gray-lighter-color  
    border-bottom-size-1" role="navigation">  
211     <div class="navbar-header">  
212       <button type="button" class="navbar-toggle collapsed"  
        data-toggle="collapse" data-target=  
        "#bs-navbar-collapse-twoStyle">  
213         <span class="sr-only">Toggle navigation</span>  
214         <span class="icon-bar"></span>  
215         <span class="icon-bar"></span>  
216         <span class="icon-bar"></span>  
217       </button>  
218       <div><h3 class="margin-level-size-4 line-height-4"><img src=  
        "images/logo01.png" width="30" height="30"> 直播课时间表  
        </h3></div>  
219     </div>
```

(1) 第 209 行,使用了自定义的.myContains-style-2 组件样式,主要是渲染第二种风格的推荐位的层叠样式。

(2) 第 210 行,使用了自定义的 bg-gray-lighter-color 和 border-bottom-size-1 全局 CSS 样式。

(3) 第 218 行,使用了自定义的 margin-level-size-4 和 line-height-4 全局 CSS 样式。

```
220   <div class="collapse navbar-collapse" id=  
    "bs-navbar-collapse-twoStyle">  
221     <div class="hidden-sm hidden-md hidden-lg border-top-size-1">  
222       <ul class="nav navbar-nav">  
223         <li><a href="#">今天 -- 09月21日</a></li>  
224         <li><a href="#">星期二 -- 09月21日</a></li>  
225         <li class="active"><a href="#">星期三 -- 09月22日  
226         <a href="#">星期四 -- 09月23日</a></li>  
227         <li><a href="#">星期五 -- 09月24日</a></li>  
228         <li><a href="#">星期六 -- 09月25日</a></li>  
229         <li><a href="#">星期日 -- 09月26日</a></li>  
230       </ul>  
231     </div>  
232   </div>  
233 </nav>
```

第 221 行,使用了自定义的 border-top-size-1 全局 CSS 样式。

```

234 <div class="panel panel-default col-sm-12 col-md-9 col-lg-7">
235   <div class="panel-heading hidden-xs pull-left fluid-width">
236     <ul class="nav navbar-nav">
237       <li><a href="#">今天 <br> 09月21日</a></li>
238       <li><a href="#">星期二 <br> 09月21日</a></li>
239       <li class="active"><a href="#">星期三 <br> 09月22日
240         </a></li>
241       <li><a href="#">星期四 <br> 09月23日</a></li>
242       <li><a href="#">星期五 <br> 09月24日</a></li>
243       <li><a href="#">星期六 <br> 09月25日</a></li>
244     </ul>
245   </div>
  </div>
  <div class="clearfix"></div>

```

第 235 行，使用了自定义的 fluid-width 全局 CSS 样式。

```

246 <table class="table">
247   <tr>
248     <td>06:00</td>
249     <td>架构师/产品经理</td>
250     <td class="text-right">143244人订阅</td>
251     <td><button type="button" class="btn btn-default">
      立即预约</button></td>
252   </tr>
253   <tr>
254     <td>06:00</td>
255     <td>架构师/产品经理</td>
256     <td class="text-right">44人订阅</td>
257     <td><button type="button" class="btn btn-default">
      立即预约</button></td>
258   </tr>
259   <tr>
260     <td>06:00</td>
261     <td>架构师/产品经理</td>
262     <td class="text-right">4人订阅</td>
263     <td><button type="button" class="btn btn-default">
      立即预约</button></td>
264   </tr>
265 </table>
266 </div>
267 <div class="panel panel-default hidden-xs hidden-sm col-md-3
col-lg-push-1 col-lg-4 right">
268   <div class="panel-heading">
269     <h3>我的订阅</h3>
270   </div>
271
272   <table class="table">
273     <tr>
274       <td>09/15 06:00</td>
275       <td>架构师/产品经理</td>
276     </tr>
277     <tr>
278       <td>09/15 06:00</td>
279       <td>架构师/产品经理</td>
280     </tr>
281     <tr>
282       <td>09/15 06:00</td>
283       <td>架构师/产品经理</td>
284     </tr>
285   </table>

```



细说网页制作

```
286 </div>
287 </section>
288 </div>
```

第 267 行，使用了自定义的 `.right` 组件样式，用来定义第二种风格的推荐位中的右侧面板。

运行结果如图 13.34～图 13.37 所示。

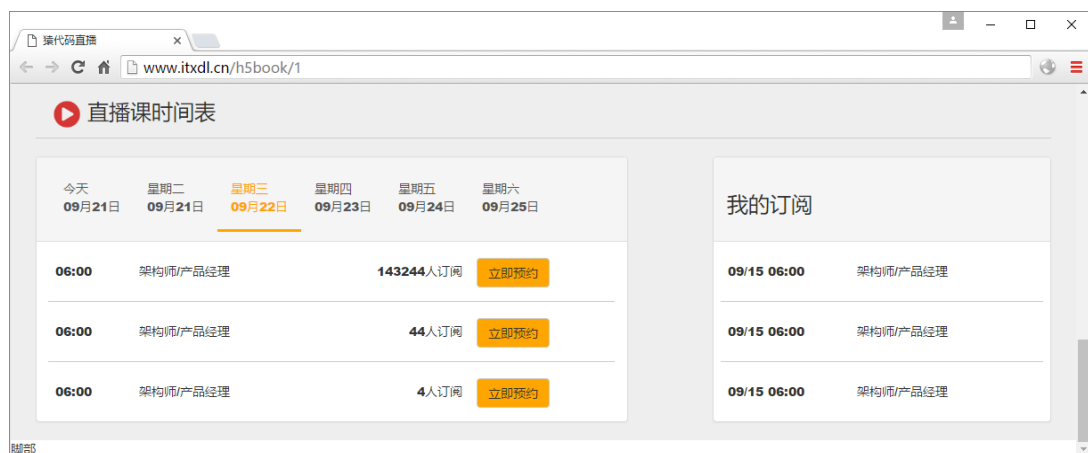


图 13.34 第二种风格的推荐位的样式优化——大屏运行结果

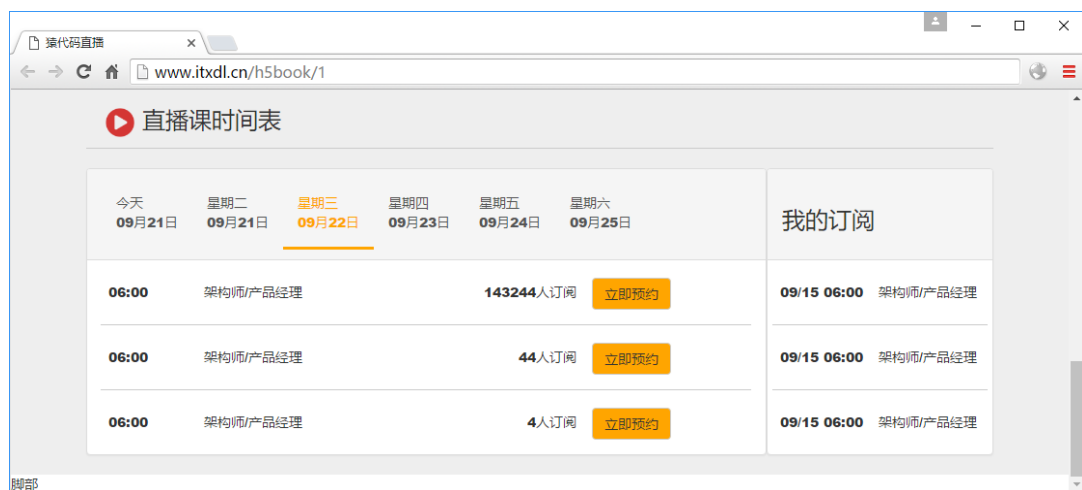


图 13.35 第二种风格的推荐位的样式优化——中等屏幕运行结果



图 13.36 第二种风格的推荐位的样式优化——
小屏运行结果



图 13.37 第二种风格的推荐位的样式
优化——超小屏运行结果

此时，两种风格的推荐位全部完成，读者可以根据实际情况自定义更多风格的推荐位。其中需要注意的有：如何实现页面适配功能、页面主次内容部分该如何进行适配等。

13.8 网站的脚部

网站的脚部主要显示本网站的基本信息、公司的联系方式、友情链接、网站备案信息、网站版权信息等。而一般移动端页面并不显示这类信息，而是直接对其隐藏。其代码如下所示。

实例代码：

```

289 <!-- 脚部 -->
290 <footer class="bg-gray-lighter-color">
291   <footer id="footer" class="container">
292     <div class="hidden-xs col-sm-5">
293       <address>
294         <strong>联系方式</strong><br><br>
295         QQ:12345678<br><br>
296         <span class="glyphicon glyphicon-map-marker"></span>
          北京市昌平区文华西路育荣教育园区兄弟连楼 (123) 123-456
297       </address>
298     </div>
299     <div class="hidden-xs col-sm-7 text-right pull-right">
300       <address>

```



细说网页制作

```
301      <br><br>
302      Copyright 易第优（北京）教育咨询股份有限公司 2006 - 2016 Edu
303      Inc.<br><br>
304      京ICP备11018177号 京公网安备11011402000177
305      </address>
306      </div>
307      </footer>
307 </footer>
```

运行结果如图 13.38 所示。



图 13.38 网站的脚部

当然，读者可以对应地丰富一些其他信息，比如友情链接等。

至此，整个页面基本大功告成。接下来，笔者再添加一个新的模块——用户登录模块，来进一步学习 Bootstrap 的表单和模态框的使用。

13.9 用户登录

用户登录少不了表单，并且模态框也是项目开发中必用的一部分，在此笔者将对这两个知识点进行进一步的讲解，并结合这两个知识点完成 PC 端用户登录模块。

13.9.1 Bootstrap的表单全局CSS样式

Bootstrap 中的表单是全局 CSS 样式，不是组件，这在上一章节中也有所提及。因为它的运用非常广泛，所以笔者也将对此进行详细讲解。

Bootstrap 的表单从布局上分为 3 种类型：基本样式表单、内联样式表单和水平排列表单。此外，每个控件还可以单独设置一些常见属性，比如被支持的控件、静态控件、焦点状态、禁用状态、只读状态、校验状态、控件尺寸和辅助文本等。

实例代码:

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>bootstrap的表单</title>
8   <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11   <form role="form">
12     <div class="form-group">
13       <label for="exampleInputEmail1">邮箱: </label>
14       <input type="email" class="form-control" id="exampleInputEmail1"
15         placeholder="请输入您的邮箱地址">
16     </div>
17     <div class="form-group">
18       <label for="exampleInputPassword1">密码</label>
19       <input type="password" class="form-control" id="
20         exampleInputPassword1" placeholder="请输入您的邮箱密码">
21     </div>
22     <div class="checkbox">
23       <label>
24         <input type="checkbox"> 记住密码
25       </label>
26     </div>
27     <button type="submit" class="btn btn-default">进入邮箱</button>
28   </form><hr>
29   <form class="form-horizontal" role="form">
30     <div class="form-group">
31       <label for="inputEmail3" class="col-sm-2 control-label">邮箱
32       <div class="col-sm-10">
33         <p class="form-control-static"> email@example.com</p>
34       </div>
35     </div>
36     <div class="form-group">
37       <label for="inputPassword3" class="col-sm-2 control-label">密码
38       <div class="col-sm-10">
39         <input type="password" class="form-control" id="inputPassword3"
40           placeholder="请输入您的邮箱密码" disabled>
41       </div>
42     </div>
43   </form>
44   <form class="form-inline" role="form">
45     <div class="form-group has-warning has-feedback">
46       <label class="sr-only" for="exampleInputEmail2">邮箱</label>
47       <input type="email" class="form-control" id="exampleInputEmail2"
48         placeholder="请输入你的邮箱地址">
49       <span class="help-block">你输入的信息是正确的</span>
50       <span class="glyphicon glyphicon-ok form-control-feedback"></span>
51     </div>
52     <div class="form-group">
53       <label class="sr-only" for="exampleInputPassword2">密码</label>

```

基础表单样式

获取更好排列的表单控件

表单控件样式类

被支持的HTML5表单控件

水平排列的表单

栅格系统排列

静态控件

禁用状态

校验状态

添加控件图标

辅助文本

添加控件图标



```
67     <input type="password" class="form-control input-lg" id=  
68         "exampleInputPassword2" placeholder="请输入你的邮箱密码">  
69 </div>  
70 <div class="checkbox">  
71     <label>  
72         <input type="checkbox"> 记住密码  
73     </label>  
74 </div>  
75 <button type="submit" class="btn btn-default">进入邮箱</button>  
76 </form>  
77 <script src="js/jquery-2.2.4.min.js"></script>  
78 <script src="js/bootstrap.js"></script>  
79 </body>  
80 </html>
```

控件大小

运行结果如图 13.39 所示。

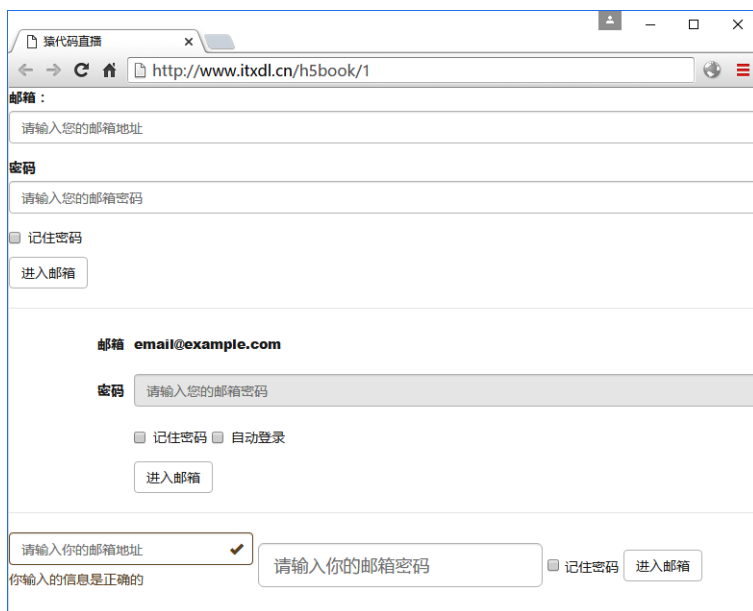


图 13.39 Bootstrap 的表单全局 CSS 样式运行结果

上述代码基本介绍了 Bootstrap 表单中的主要知识点，而实际用法还需要读者根据实际情况灵活变通。

13.9.2 Bootstrap的模态框JavaScript插件

Bootstrap 的模态框使用起来非常便捷，分为三部分：头部、内容部分和脚部。此外，还可以调整模态框的大小和出现的动画效果。下面我们就来看看其实现代码。

实例代码:

```

1 <!DOCTYPE html>
2 <html lang="zh-CN">
3 <head>
4   <meta charset="utf-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>bootstrap的模态框</title>
8   <link href="css/mybootstrap.css" rel="stylesheet">
9 </head>
10 <body>
11 <!-- Button trigger modal -->
12 <button type="button" class="btn btn-primary btn-lg" data-toggle="modal"
13   data-target="#myModal">
14   Launch demo modal
15 </button>
16 <!-- Modal -->
17 <div class="modal fade" id="myModal" tabindex="-1" role="dialog"
18   aria-labelledby="myModalLabel" aria-hidden="true">
19   <div class="modal-dialog modal-sm">
20     <div class="modal-content">
21       <div class="modal-header">
22         <button type="button" class="close" data-dismiss="modal"><span
23           aria-hidden="true">&times;</span><span class="sr-only">Close
24         </span></button>
25       <h4 class="modal-title" id="myModalLabel">Modal title</h4>
26     </div>
27     <div class="modal-body">
28       ...
29     </div>
30     <div class="modal-footer">
31       <button type="button" class="btn btn-default" data-dismiss="modal">
32         Close</button>
33       <button type="button" class="btn btn-primary">Save changes</button>
34     </div>
35   </div>
36 </div>
37 <script src="js/jquery-2.2.4.min.js"></script>
38 <script src="js/bootstrap.js"></script>
39 </body>
40 </html>

```

淡入淡出的动画效果

尺寸大小

模态框的头部

模态框的内容部分

模态框的脚部

运行结果如图 13.40 所示。

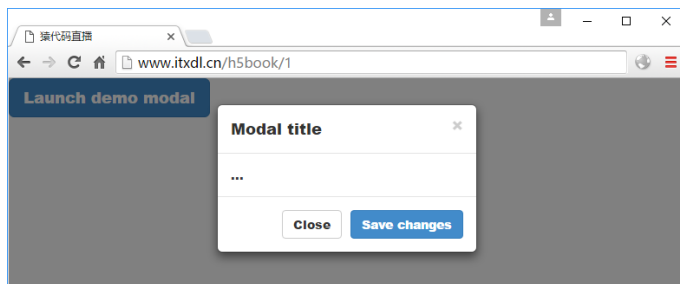
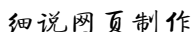


图 13.40 Bootstrap 的模态框 JavaScript 插件运行结果



可以发现，Bootstrap 的模态框运用也非常简单、易懂。

13.9.3 实战用户登录

学习完上述 Bootstrap 中的表单和模态框,现在笔者结合两者来完成实战项目中 PC 端的用户登录模块。

实例代码:

[illegible]

第 84 行，`href` 属性绑定了锚点，并添加了 `data-toggle="modal"`，目的是为“登录”超链接绑定模态框触发按钮。

```

308 <!-- 登录的模态框-->
309 <div class="modal fade model-style-1" id="login-model" tabindex="-1" role=
"dialog" aria-labelledby="myModalLabel" aria-hidden="true" >
310   <div class="modal-dialog">
311     <div class="modal-content">
312       <div class="modal-header">
313         <button type="button" class="close" data-dismiss="modal">
314           <span aria-hidden="true">&times;</span>
315           <span class="sr-only">关闭</span>
316         </button>
317         <h4 class="modal-title" id="myModalLabel">登录</h4>
318       </div>
319       <div class="modal-body">
320         <form class="form-horizontal" role="form">
321           <div class="form-group">
322             <label for="inputEmail3" class="col-sm-6
control-label">账号&nbsp;&nbsp;&nbsp;</label>
323             <div class="col-sm-5">
324               <input type="email" class="form-control" id=
"inputEmail3" placeholder="账号">
325             </div>
326           </div>

```

第 309 行，使用自定义的 `.model-style-1` 组件样式，为用户登录模块渲染独有的层叠样式。

```
327         <div class="form-group">
328             <label for="inputPassword3" class="col-sm-6
              control-label">密码</label>
329             <div class="col-sm-5">
330                 <input type="password" class="form-control" id=
                  "inputPassword3" placeholder="密码">
```

```

331         </div>
332     </div>
333     <div class="form-group">
334         <div class="col-sm-offset-6 col-sm-5">
335             <div class="checkbox">
336                 <label>
337                     <input type="checkbox"> 记住我
338                 </label>
339             </div>
340         </div>
341     </div>
342     <div class="form-group">
343         <div class="col-sm-offset-6 col-sm-10">
344             <button type="submit" class="btn btn-default">
                登录</button>
345         </div>
346     </div>
347 </form>
348 </div>
349 <div class="modal-footer">
350     <label for="" class="col-sm-7 control-label">其他方式登录
351     </label>
352     <ul class="nav nav-pills" role="tablist">
353         <li role="presentation" class="active"></li>
355         <li role="presentation" class="active"></li>
356     </ul>
357 </div>
358 </div>

```

运行上述代码，结果如图 13.41 所示。



图 13.41 实战用户登录的运行结果

可以发现，以上运行页面存在一些缺陷：第一，模态框太靠上而不美观；第二，输入框组的表单控件堆叠顺序过大（ $z\text{-index}=2$ ）。接下来对其样式进行微调，如下所示。



Sass 实战代码 (mybootstrap/_mystyle.scss):

```
120 //更改某些Bootstrap的内部样式
121 //修改模态框的偏移量
122 .model-style-1{
123     top:20%;
124 }
125 //修改遮盖层的堆叠顺序
126 .modal-backdrop {
127     z-index: 3;
128 }
```

运行结果如图 13.42 所示。

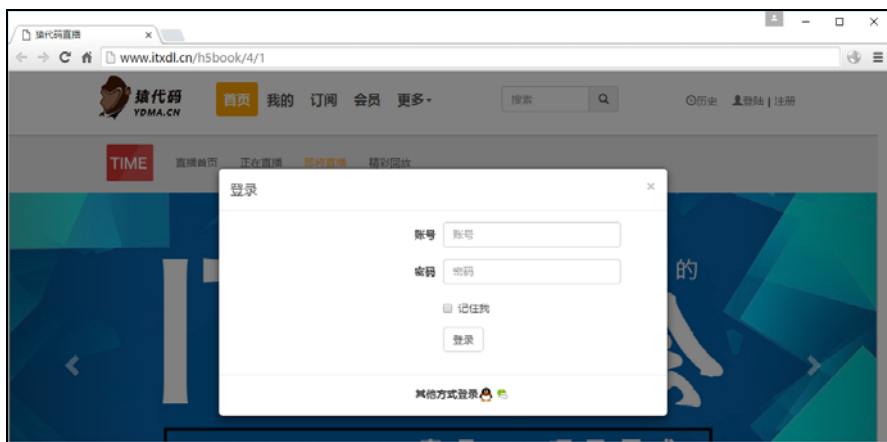


图 13.42 实战用户登录的样式微调后的运行结果

至此，整个项目已全部完成，但 Bootstrap 的运用是一个熟能生巧的过程，大家必须多加练习，才能更好地理解和灵活运用 Bootstrap。Sass 的学习也是如此。

本章小结

本章带领大家从零开始完成一个项目，按照项目需求来驱动项目开发，并从中学习 Bootstrap 和简单的 Sass 运用。本章详细知识点如下：

- 首先，简单介绍了本次实战的项目需求，把整个页面分为五个功能模块，然后对应其功能模块进行具体细节细分，准备好 Sass 配置和 HTML 的模块划分。
- 其次，从顶部导航到脚部一一讲解，首先讲解对应功能模块的具体实现思路，然后学习其需要的 Bootstrap 知识点，最后完成项目实战。
- 最后，又补充了一个用户登录模块，学习了两个常用的 Bootstrap 知识点——表单全局 CSS 样式和模态框。

本章习题

- 下面对 Bootstrap 的栅格系统描述错误的是（ ）。
 - Bootstrap 栅格系统默认分为 12 栏，当然也可以修改默认配置
 - Bootstrap 栅格系统可以适配四种不同宽度的屏幕
 - Bootstrap 栅格系统可以按栏进行偏移
 - Bootstrap 栅格系统可以按行进行排序
- 下面对 Bootstrap 的表格描述错误的是（ ）。
 - 条纹状表格使用的是 .table-striped 类
 - 带边框的表格使用的是 .table-bordered 类
 - 表格的状态类有 .active、.success、.primary、.info、.warning 和 .danger 六大类
 - 响应式表格会在小屏幕设备上（小于 768px）水平滚动，如果大于 768px，则水平滚动条会消失
- 下面对 Bootstrap 的导航条叙述错误的是（ ）。
 - Bootstrap 导航条是继承于导航组件之上的
 - Bootstrap 导航条默认可以设置情景颜色
 - Bootstrap 导航条可以嵌套下拉菜单组件、按钮组、输入框组等
 - Bootstrap 导航条可以悬浮在窗口的顶部或底部
- 下面哪个不是 Bootstrap 的组件？【多选】（ ）
 - 进度条
 - 轮播图
 - 警告框
 - 巨幕
- 下面对 Bootstrap 的表单叙述不正确的是（ ）。
 - 表单主要分为基本样式表单、内联样式表单、水平排列表单三种类型
 - 表单可以使用 <p> 标签设置静态控件
 - 表单使用的校验状态类有 .has-warning、.has-error 和 .has-success 三类。
 - 表单组属于 Bootstrap 的全局 CSS 范畴



本章习题及其答案



本章资源包



本章扩展知识

附录A

HTML5 新增标签元素

表 A.1 HTML5 新增的结构元素及说明

元 素	说 明
section	表示页面中的一个内容区块，比如章节、页眉、页脚或页面中的其他部分。它可以与 h1、h2、h3、h4、h5、h6 等元素结合起来使用标记文档结构
article	表示页面中的一块与上下文不相关的独立内容，如博客中的一篇文章
aside	表示 article 元素内容之外的，与 article 元素内容相关的辅助信息
header	表示页面中一个内容区块或整个页面的标题
hgroup	用于整个页面或页面中一个内容区块的标题进行组合
footer	表示整个页面或页面中一个内容区块的脚注。一般来说，它会包含创作者的姓名、创建日期及创作者的联系信息
nav	表示页面中导航链接的部分
figure	表示一段独立的流内容，一般表示文档主体流内容中的一个独立单元，使用 figcaption 元素为 figure 元素组添加标题

表 A.2 HTML5 新增的多媒体元素及说明

元 素	说 明
audio	定义声音，比如音乐或其他音频流
video	定义视频，比如电影片段或其他视频流

表 A.3 HTML5 中增加的其他元素

元 素	说 明
mark	表示页面中需要突出显示或高亮显示的、对于当前用户具有参考作用的一段文字。它通常用于引用原文的时候，目的是吸引当前用户的注意。例如，对网页全文检索某个关键词时显示的检索结果 <code><mark>HTML5 内容</mark></code>
wbr	该元素表示软换行，br 元素表示此处必须换行，而 wbr 元素的意思是当浏览器窗口或父级元素的宽度足够宽时（没有必要换行时），不进行换行；而当宽度不够时，主动在此处进行换行。但是对于中文没有太大用处
progress	定义任何类型的任务的进度。可以使用<progress>标签来显示 JavaScript 中耗时间的函数的进度
meter	定义预定义范围内的度量。仅用于已知最大值和最小值的度量
canvas	用于定义图形，绘制路径，像矩形、圆形、字符及添加图像的方法。canvas 元素本身没有绘图能力，所有的绘制工作必须在 JavaScript 内部完成
command	表示命令按钮，如单选按钮、复选框或按钮 <code><command onclick="cut()" label="cut"></code>
details	表示用户要求得到并且可以得到的细节信息，即定义元素的细节
ruby	该标签定义注释（中文注音或字符）在东亚使用

HTML5 新增表单元素和属性

表 A.4 HTML5 中新增表单元素

元 素	说 明
datalist	规定输入域的选项列表
keygen	标签提供了一种验证用户的可靠方法。keygen 元素是密钥对生成器（Key-pair Generator）。当提交表单时，会生成两个键，一个是私钥，另一个是公钥。私钥（Private Key）存储于客户端，公钥（Public Key）则被发送到服务器。公钥可用于之后验证用户的客户端证书（Client Certificate）
output	用于不同类型的输出，比如计算或脚本输出

表 A.5 HTML5 新增 input 类型

类 型	说 明
email	用于包含 E-mail 地址的输入域
url	用于包含 URL 地址的输入域
number	用于包含数值的输入域



续表

类 型	说 明
range	用于应该包含一定范围内数字值的输入域，range 类型显示为滑动条
date picker	日期选择器，可选 date、month、week、time、datetime、datetime-local
search	用于搜索域
color	用于选取颜色

表 A.6 HTML5 新增表单及 input 属性

属 性	说 明
autocomplete	规定 form 或 input 域应该拥有自动完成功能
autofocus	规定在页面加载时，域自动获得焦点
form	规定输入域所属的一个或多个表单
form overrides	表单重写属性，允许重写 form 元素的某些属性设定，有以下属性：formaction、formenctype、formmethod、formnovalidate、formtarget
height and width	规定用于 image 类型的<input>标签的图像高度和宽度
list	规定输入域的 datalist
min, max	用于包含数字或日期的 input 类型规定限定（约束）
step	为输入域规定合法的数字间隔
multiple	规定<input>标签可以选择多个值
novalidate	规定在提交表单时不应该验证 form 或 input 域
pattern	该属性描述一个正则表达式，用于验证<input>标签的值
placeholder	该属性提供一种提示（hint），描述输入域所期待的值
required	规定必须在提交之前填写输入域（不能为空）

表 A.7 表单重写属性

属 性	描 述
formaction	重写表单的 action 属性，用于描述表单提交的 URL 地址
formenctype	重写表单的 enctype 属性，用于描述表单提交到服务器的数据编码（只对 form 表单中 method="post" 表单）
formmethod	重写表单的 method 属性，定义了表单提交的方式，如 get、post

续表

属 性	描 述
formnovalidate	重写表单的 novalidate 属性，描述了<input>标签在表单提交时无须验证
formtarget	重写表单的 target 属性，指定一个名称或一个关键字来指明表单提交数据接收后的展示，如_blank

HTML5 废除的标签元素

表 A.8 在 HTML5 中被废除的属性

在 HTML4.0 中使用的属性	使用该属性的元素	在 HTML5 中的替代方案
rev	link、a	rel
charset	link、a	在被链接的资源中使用 HTTP Content-type 头元素
shape、coords	a	使用 area 元素替代 a 元素
longdesc	img、iframe	使用 a 元素链接到较长描述
target	link	多余属性，被省略
nohref	area	多余属性，被省略
profile	head	多余属性，被省略
version	html	多余属性，被省略
name	img	id
scheme	meta	只为某个表单域使用 scheme
archive、chlassid、codebase、codetype、declare、standby	object	使用 data 与 type 属性类调用插件。需要使用这些属性来设置参数时，使用 param 属性
valuetype、type	param	使用 name 与 value 属性规定值的 MIME 类型
axis、abbr	td、th	使用以明确简洁的文字开头、后跟详述文字的形式。可以对更详细的内容使用 title 属性，以使单元格的内容变得简短
scope	td	在被链接的资源中使用 HTTP Content-type 头元素
align	caption、input、legend、div、h1、h2、h3、h4、h5、h6、p	使用 CSS 样式表替代
alink、link、text、vlink、background、bgcolor	Body	使用 CSS 样式表替代



续表

在 HTML4.0 中使用的属性	使用该属性的元素	在 HTML5 中的替代方案
align、bgcolor、border、cellpadding、cellspacing、frame、rules、width	table	使用 CSS 样式表替代
align、char、charoff、height、nowrap、valign	tbody、thead、tfoot	使用 CSS 样式表替代
align、bgcolor、char、charoff、height、nowrap、valign、width	td、th	使用 CSS 样式表替代
align、bgcolor、char、charoff、valign	tr	使用 CSS 样式表替代
align、char、charoff、valign、width	col、colgroup	使用 CSS 样式表替代
align、border、hspace、vspace	object	使用 CSS 样式表替代
clear	br	使用 CSS 样式表替代
compact、type	ol、ul、li	使用 CSS 样式表替代
compact	dl	使用 CSS 样式表替代
compact	menu	使用 CSS 样式表替代
width	pre	使用 CSS 样式表替代
align、hspace、vspace	img	使用 CSS 样式表替代
align、noshade、size、width	hr	使用 CSS 样式表替代
align、frameborder、scrolling、marginheight、marginwidth	Iframe	使用 CSS 样式表替代
autosubmit	menu	